UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

INFERÊNCIA DE ATIVIDADES CLÍNICAS NA ARQUITETURA CLINICSPACE A PARTIR DE PROPRIEDADES DO CONTEXTO

DISSERTAÇÃO DE MESTRADO

Marcos Vinícius Bittencourt de Souza

Santa Maria, RS, Brasil
2010

INFERÊNCIA DE ATIVIDADES CLÍNICAS NA ARQUITETURA CLINICSPACE A PARTIR DE PROPRIEDADES DO CONTEXTO

por

Marcos Vinícius Bittencourt de Souza

Dissertação apresentada ao Curso de Mestrado em Computação do Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Computação**

Orientadora: Prof.ª Iara Augustin

Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

INFERÊNCIA DE ATIVIDADES CLÍNICAS NA ARQUITETURA CLINICSPACE A PARTIR DE PROPRIEDADES DO CONTEXTO

elaborada por
Marcos Vinícius Bittencourt de Souza

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Iara Augustin, Dra.
(Presidente/Orientadora)

Adenauer Correa Yamin, Dr. (UFPEL)

Lisandra Manzoni Fontoura, Dra. (UFSM)

Santa Maria, 10 de setembro de 2010.

AGRADECIMENTOS

Agradeço a Deus e seus bem-feitores pela orientação nos caminhos trilhados até aqui.

Agradeço à minha orientadora por me aceitar como orientando e, com isso, me proporcionar o acesso a um aprendizado de qualidade, motivador e dinâmico presente no Programa de Pós-Graduação em Informática da UFSM. O seu apoio e conhecimento foram muito necessários no presente trabalho e ajudaram a superar as dificuldades encontradas.

Agradeço à minha noiva por compreender os momentos de ausência devido às aulas e horas necessárias para a realização dessa pesquisa. Com certeza, o seu apoio foi fundamental para essa realização. À sua família por sempre incentivar a superar os obstáculos.

Agradeço à minha família por mais essa conquista realizada e pelo eterno ‘voto de confiança’ depositado a mim. Sempre estiveram e estarão presentes nos momentos difíceis amenizando-os e oferecendo conforto. O incentivo desde cedo à educação e busca do conhecimento permitiu a realização de mais essa etapa.

Agradeço aos amigos e colegas de trabalho pelo constante incentivo e força de vontade para que a conclusão desse trabalho se realizasse.

Obrigado a todos os demais envolvidos direta ou indiretamente na realização desse trabalho, com a certeza de que todas as colaborações foram úteis de alguma forma.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

INFERÊNCIA DE ATIVIDADES CLÍNICAS NA ARQUITETURA CLINICSPACE A PARTIR DE PROPRIEDADES DO CONTEXTO

Autor: Marcos Vinícius Bittencourt de Souza
Orientadora: Iara Augustin
Data e Local da Defesa: Santa Maria, 10 de setembro de 2010

De forma a melhorar a usabilidade do sistema e auxiliar o usuário durante a execução de suas tarefas clínicas cotidianas foram projetados novos componentes e serviços para realizar a inferência de tarefas na arquitetura ClinicSpace. O projeto ClinicSpace, em desenvolvimento no GMob do PPGI/UFSM, visa construir uma ferramenta-piloto que permita a programação das atividades clínicas pelo próprio médico e o gerenciamento automático destas. Modelar e implementar um serviço de inferência para essa arquitetura é o objetivo principal do trabalho descrito nesta dissertação. Como base para a inferência das tarefas de cada usuário, utilizou-se o histórico de execução de tarefas juntamente com as características presentes no ambiente durante a execução. Dessa forma, é possível traçar o perfil de cada usuário, conhecendo-se quais funcionalidades serão necessárias a ele no futuro. A partir da captura de informações do ambiente durante a execução das tarefas, foi utilizado o algoritmo C4.5 para inferir, prever, a próxima tarefa a ser executada. Em conjunto com o constante monitoramento do ambiente são detectados os padrões que levam a supor a execução futura de uma tarefa, auxiliando na utilização do sistema. A partir da tarefa inferida, o sistema a apresenta como sugestão na interface gráfica de forma a não tomar decisões automáticas em substituição ao papel do usuário, prevendo-se uma melhoria geral na usabilidade do sistema. Como forma de validação da arquitetura desenvolvida foram feitas análises de desempenho do mecanismo de inferência, obtendo-se o resultado de baixa interferência nos tempos de execução do sistema como um todo.

Palavras-chave: computação pervasiva; computação ubíqua; *middleware*; computação orientada a atividades; tarefas clínicas; gerenciamento de tarefas.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

Clinical Activities Inference in the ClinicSpace Architecture using Context Properties

Author: Marcos Vinícius Bittencourt de Souza
Advisor: Iara Augustin
Santa Maria, September 10, 2010

To improve the system usability and assist the user during the execution of their daily clinical tasks, were designed new components and services to realize the task inference in the ClinicSpace architecture. The ClinicSpace project, currently being developed by the GMob of the PPGI/UFSM, aims to build a pilot tool that allows the modeling of the clinical tasks by the physician and their automatic management. To model and develop an inference service to this architecture is the main goal of the work described in the current dissertation. To realize the task inference, were used the task execution history of each user together with the present characteristics of the environment during the tasks executions. In this way, is possible to trace the profile of each user, knowing which functionalities will be necessary for him in the near future. With the capture of the environment information during the task execution, was used the C4.5 algorithm to infer, foresee, the next task to be executed. Together with the constant environment monitoring, were detected patterns that allows suppose the future execution of a task, helping the system utilization. The system presents the inferred tasks as suggestion in the graphical interface to not take automatic decisions, taking the user role, predicting a general improvement in the system usability. To validate the developed architecture, were made performance analyses of the inference mechanism, resulting in a small interference of the execution time of the whole system.

Key-words: pervasive computing; ubiquitous computing; middleware; task-driven computing; clinical activities; task management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Program Interface*
AVU	Ambiente Virtual do Usuário
BDA	Base de Dados pervasiva das Aplicações
EHS	*Electronic Healthcare System*
IETC	Interface de Edição de Tarefas e Contexto
EXEHDA	*Execution Environment for Highly Distributed Applications*
ORM	*Object-Relational Mapping*
pEHS	*pervasive EHS*
SAT	Serviço de Acesso a Tarefas
STA	Serviço de Tarefas Ativas
SCT	Serviço de Contexto de Tarefas
SGDT	Subsistema de Gerenciamento Distribuído de Tarefas
SGT	Serviço de Gerenciamento de Tarefas
SHE	Serviço de Histórico de Execução
SPT	Serviço de Predição/Inferência de Tarefas

LISTA DE APÊNDICES

SUMÁRIO

1 INTRODUÇÃO

No mundo atual, os serviços de computação são extremamente necessários e inerentes à vida das pessoas. Serviços como mensagens instantâneas, e-mails, busca de conteúdo na internet, etc, são bastante usados na rotina de vida das pessoas, fazendo com que o acesso a eles seja bastante importante.

A disponibilização de tais serviços, de uma forma transparente e com grande disponibilidade, requer meios eficazes para que o acesso do usuário seja favorecido. Contrastando com o uso tradicional, no qual o usuário necessita de um lugar fisicamente imóvel para a interação com os serviços e *softwares* existentes, a Computação Pervasiva pode utilizar dispositivos móveis para a execução de aplicações ou, ainda, migrar as aplicações em execução para o dispositivo mais próximo do usuário.

Em um ambiente hospitalar, a Computação Pervasiva permite que as informações necessárias ao corpo médico sejam acessíveis de todos os lugares, facilitando a realização de suas tarefas cotidianas. Dessa forma, os dados do sistema ficam espalhados e podem ser acessados e manipulados em diversas localizações, estando presentes quando é necessária a entrada de informações no sistema.

As atividades exercidas por um médico podem ser descritas como um conjunto de rotinas que são executadas, na grande maioria das vezes, da mesma forma. Essas atividades são decompostas em subatividades que contêm os passos a serem executados para a realização de suas tarefas formando um fluxo de trabalho (*workflow*). Conforme a natureza das atividades clínicas, elas podem ser executadas por diferentes pessoas de forma colaborativa, fazendo com que o sistema informatizado tenha esse suporte.

Acredita-se que os sistemas de execução dos *workflows* clínicos possam ter sua a usabilidade melhorada caso sejam sensíveis ao contexto. Nesse caso, de acordo com a rotina do médico, o sistema pode agir pró-ativamente, sugerindo as aplicações a serem usadas e fornecendo alguns dados recolhidos do ambiente. Por exemplo, antes de o médico realizar a consulta de um determinado paciente, o sistema pode identificar qual o paciente em questão e fornecer as informações necessárias à atividade. Mesmo com esse comportamento, essas informações devem ser tratadas como sugestões do sistema, podendo ser alteradas de acordo

com a necessidade. Assim, o usuário mantém o controle do sistema.

1.1 Motivação e Escopo do Trabalho

No âmbito de sistemas pervasivos de saúde, o Grupo de Pesquisa em Sistemas de Computação Móvel da UFSM está desenvolvendo o projeto ClinicSpace (FERREIRA, 2009a) (FERREIRA, 2009b) (MACHADO, 2010) (SILVA, 2009a) (SILVA, 2009b) (VICENTINI, 2010) , que visa construir uma arquitetura pervasiva voltada a atividades humanas incluindo uma ferramenta-piloto para a modelagem e personalização das atividades clínicas (realizada pelo próprio usuário), associando-as a elementos de contexto. O projeto ClinicSpace é o escopo de desenvolvimento dessa dissertação. A arquitetura ClinicSpace é modelada em serviços; dentre estes, tem-se serviços para a execução e o gerenciamento das tarefas e, para aumentar a usabilidade desses, propõe-se nesse trabalho inserir a inferência de tarefas de acordo com o estado dos elementos de interesse do contexto, permitindo a sugestão de tarefas a executar ao usuário. O gerenciamento das aplicações do ClinicSpace é realizado com auxílio do monitoramento dos elementos de contexto, através do *middleware* EXEHDA (*Exectution Environment for Highly Distributed Applications* – Ambiente de Execução para Aplicações Altamente Distribuídas) (AUGUSTIN, 2008).

Na arquitetura ClinicSpace, ainda, está sendo adicionado um Sistema de Informação de Saúde Pervasivo – (pEHS – *Pervasive Eletronic Health System*) (VICENTINI, 2010), para dar suporte ao usuário clínico na realização das suas atividades modeladas. Para integrar a arquitetura ClinicSpace com o sistema pEHS, deve haver uma associação de funcionalidades do sistema através de subtarefas disponíveis no ClinicSpace, fazendo com que seja possível a construção das tarefas de maneira flexível. Como exemplo de funcionalidades do sistema pEHS que podem ser transformadas em subtarefas, estão: a) identificar paciente; b) buscar exames do paciente, e c) prescrever medicamento (SILVA, 2009a).

A partir das subtarefas disponíveis na ferramenta de edição de tarefas (SILVA, 2009a), associadas às funcionalidades do pEHS, o usuário pode construir as tarefas personalizadas a serem realizadas no seu cotidiano. O fluxo de trabalho do clínico forma uma tarefa composta de diversas subtarefas, dispostas de acordo com a maneira que ele as executa no seu ambiente de trabalho. Após a modelagem das tarefas, o clínico pode usá-las para formar novas tarefas, de forma a agrupá-las e reaproveitar o máximo de trabalho, permitindo um melhor

gerenciamento.

A ligação das subtarefas com os elementos de contexto leva em consideração os aspectos que permitem uma captação de dados para o preenchimento de informações na aplicação pEHS (RIZETTI, 2009b). A partir dessa configuração, a captação das informações do contexto aumenta a usabilidade das aplicações do pEHS. Essa característica faz com que o usuário otimize o seu tempo, informando dados mais importantes enquanto as informações do contexto são automaticamente fornecidas.

A modelagem das tarefas permite, ainda, que sejam informadas condições que ativam automaticamente determinadas aplicações do pEHS, de acordo com o estado dos elementos do contexto. Dessa maneira, deve haver um constante monitoramento de tais informações para que a ativação da tarefa ocorra. Exemplificando, o clínico pode modelar a tarefa e configurar para que, a uma determinada hora do dia, seja executada uma aplicação, ou ainda, que quando for detectado um paciente no ambiente a sua ficha de informações seja exibida.

1.2 Objetivo do Trabalho

Conforme a complexidade de um sistema aumenta e várias funcionalidades lhe são adicionadas, pode-se correr o risco de tornar a interface gráfica do sistema confusa, levando o usuário a tomar decisões erradas. Em alguns casos, pode ser necessária a execução de uma tarefa de emergência e, caso o sistema não tenha uma interface bem estruturada, pode levar o usuário a executar tarefas indesejadas ou fazer com que ele não utilize a ferramenta. A grande quantidade de informações presente na interface e a sua má organização levam à sobrecarga cognitiva no usuário, o que diminui a sua eficiência na utilização do sistema como um todo (CHANDLER, 1991).

A conjunção de tais fatores, muitas vezes, leva ao abandono ou rejeição do sistema, devido à complexidade implícita para a realização de tarefas cotidianas. Como reflexo do mau uso do sistema, pode-se ter informações incorretas e imprecisas, fazendo com que o sistema não seja confiável.

De forma a melhorar a usabilidade do sistema e auxiliar o usuário durante a execução de suas tarefas clínicas cotidianas, foram projetados novos componentes e serviços para realizar a inferência de tarefas na arquitetura ClinicSpace. Modelar e implementar um serviço de inferência para a arquitetura é o objetivo principal desse trabalho, descrito nesta

dissertação.

Como base para a inferência das tarefas de cada usuário, utilizou-se o histórico de execução de tarefas juntamente com as características presentes no ambiente durante a execução. Dessa forma, é possível traçar o perfil de cada usuário, conhecendo-se quais funcionalidades serão necessárias a ele no futuro.

A partir da captura de informações do ambiente durante a execução das tarefas, é utilizado o algoritmo C4.5 para inferir, prever, a próxima tarefa a ser executada. Em conjunto com o constante monitoramento do ambiente, são detectados os padrões que levam a supor a execução futura de uma tarefa, auxiliando na utilização do sistema.

Para diminuir o impacto da utilização de um sistema informatizado, o sistema não executa automaticamente as tarefas inferidas, mas sim, as apresenta como forma de sugestão. Dessa maneira, o usuário permanece sempre com o controle total do sistema e é responsável pela tomada de decisões, permanecendo o sistema apenas como meio de realização de suas tarefas e não como o seu executor.

1.3 Organização do Texto

Este texto está organizado conforme sequência a seguir. O Capítulo 2 apresenta principais conceitos da Computação Pervasiva e suas peculiaridades, e exibe a arquitetura do *middleware* EXEHDA que dá o suporte de gerenciamento do ambiente pervasivo utilizado no presente trabalho. O Capítulo 3 detalha a arquitetura ClinicSpace para a qual o serviço de inferência foi desenvolvido. O Capítulo 4 exibe as principais características e funcionalidades levadas em consideração para a realização de inferência de tarefas e detalha a arquitetura de componentes desenvolvida nessa dissertação, assim como a sua integração com os demais componentes presentes na arquitetura ClinicSpace. O Capítulo 5 apresenta a análise dos resultados obtidos, bem como os trabalhos relacionados à inferência de tarefas, foco principal dessa dissertação. O Capítulo 6 apresenta as conclusões obtidas após a realização do trabalho, assim como apresenta sugestões de trabalhos futuros a serem desenvolvidos para complementar o desenvolvimento realizado.

2 COMPUTAÇÃO PERVASIVA

Neste capítulo são apresentadas as principais características e conceitos envolvidos na Computação Pervasiva. São abordados o funcionamento e comportamento das aplicações pervasivas, bem como das aplicações conscientes de contexto.

2.1 Características das Aplicações Pervasivas

De uma forma geral, a Computação Pervasiva trata do acesso à computação de qualquer lugar ("*anywhere*"), a qualquer momento ("*anytime*") e de qualquer dispositivo ("*any device*"), de forma transparente ao usuário. Com isso, os dispositivos que cercam as pessoas têm a capacidade de trocar informações entre si para prover serviços, levando, muitas vezes, à sensação de desconhecimento dessa computação (WEISER, 1993).

A Computação Pervasiva remete a um ambiente no qual os serviços, dispositivos e recursos, em geral, descobrem-se uns aos outros e integram-se provendo um serviço útil ao usuário de forma transparente, sem a interferência direta dele (COUTAZ, 2008). Nesse cenário, um dos principais desafios é inferir as atividades do usuário pró-ativamente e, a partir disso, identificar os recursos disponíveis necessários para a realização de tais atividades. Comparados com os sistemas tradicionais, nos quais o usuário deve identificar e alocar manualmente os recursos, as aplicações pervasivas podem ter meios para descobrir quais recursos serão necessários em um futuro próximo e já disponibilizá-los ao usuário. Como conseqüência, o desenvolvimento de tais aplicações tende a se tornar diferente do desenvolvimento de aplicações tradicionais uma vez que pode/deve reagir ao ambiente no qual estão inseridas.

Do ponto de vista do desenvolvimento de aplicações pervasivas, existem outras características importantes a serem levadas em consideração e que podem ser difíceis de serem implementadas: interface natural, conhecimento do contexto e adaptação ao meio (ABOWD, 2000). A primeira característica diz respeito à forma como as pessoas interagem

com as aplicações, permitindo que essa interação possa ser feita com comandos de voz, gestos, etc, como é a forma de comunicação entre as pessoas.

Sistemas que possuam tais interfaces, ditas naturais, tendem a se sobressair às aplicações tradicionais, que utilizam somente teclado, mouse e tela como meio de interação, pois necessitam um tempo menor de aprendizado do usuário e se tornam mais atrativas ao uso. Além disso, podem permitir que pessoas com necessidades especiais utilizem o sistema. Tais aspectos influenciam positivamente na usabilidade do sistema e os torna mais adequadas ao cotidiano das pessoas (ABOWD, 2000).

A segunda característica refere-se à capacidade de a aplicação conhecer e reagir ao meio no qual está inserida (contexto), comunicando-se com outros dispositivos quando necessário. Essa descoberta pode ter o intuito de distribuir o processamento de tarefas complexas ou recolher informações automaticamente do ambiente. Com isso, pode-se perceber a heterogeneidade do ambiente, no qual dispositivos de diferentes naturezas se agregam para prover um mesmo serviço (KALAPRIVA, 2004).

A utilização dos elementos do contexto, para modificar o comportamento das aplicações caracteriza a Computação Consciente de Contexto (*context-aware computing*) – um aspecto necessário para caracterizar aplicações como pertencentes à Computação Pervasiva. A partir do contexto, diversas adaptações podem ser realizadas, visando uma melhor usabilidade, coleta de informações ou uso de dispositivos “inteligentes”.

Em um cenário-exemplo, pode-se imaginar que uma determinada pessoa está enviando seus e-mails em um dispositivo móvel, quando a conectividade da rede sem fio piora. A aplicação consciente do contexto pode resolver enviar as mensagens com maior prioridade antes das demais, de modo a aproveitar a conectividade que pode não ser mais disponível em instantes.

2.2 Computação Consciente de Contexto

> Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é considerado relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação em si (DEY, 2001, p. 5).

Entende-se o contexto como sendo um conjunto de atributos das entidades que

compõem o ambiente em um determinado momento. Para Yamin (2004, p. 56), contexto é "Toda informação relevante para a aplicação e que pode ser obtida por esta" sendo composto por entidades que coletam informações para as quais foram projetadas e as fornecem para a aplicação ou *middleware* no qual estão inseridas. De acordo com a necessidade de monitoramento do ambiente, são usadas entidades diversas cujos dados são obtidos normalmente por meio de sensores, para cada atributo de interesse da aplicação.

A situação dos atributos das entidades presentes no ambiente pode informar a situação de objetos, pessoas, dispositivos, etc. Como exemplos de atributos, podem-se citar: localização geográfica, conectividade e latência de rede, tipo de dispositivo usado, perfil do usuário, softwares disponíveis, etc. (SCHILIT, 1994).

De acordo com a temporalidade, as informações do contexto são classificadas como dinâmicas ou estáticas. Geralmente, as informações estáticas dizem respeito às características dos dispositivos, como, por exemplo, poder de processamento ou resolução de tela, sendo, em geral, de acesso mais direto e fácil. Em contrapartida, as informações dinâmicas, como carga de processamento ou localização geográfica, podem ser mais custosas de serem adquiridas devido à sua natureza variável.

A coleta das informações do contexto pode envolver recursos dedicados ou características dos próprios dispositivos que executam a aplicação. Para se saber o posicionamento de um determinado objeto, pode-se usar um aparelho GPS; já a carga do processador ou a resolução da tela do dispositivo que executa a aplicação pode ser informada por ele mesmo. Dessa forma, tem-se uma grande variedade de formas de aquisição dos atributos das entidades que formam o contexto, o que pode contribuir para aumentar a complexidade no desenvolvimento desse tipo de aplicação.

Para que uma determinada aplicação possa utilizar os elementos do contexto ela deve ser programada para acessar diretamente os elementos de contexto ou fazer uso de um *middleware* que disponibilize tais informações. Dessa forma, a aplicação pode interpretar as informações disponibilizadas e adaptar-se ao ambiente no qual está inserida. Para prover os aspectos necessários ao gerenciamento de um ambiente e suas aplicações pervasivas, este trabalho utiliza os serviços do *middleware* EXEHDA (YAMIN, 2004), aproveitando os componentes já existes e desenvolvidos para esse fim.

2.3 *Middleware* EXEHDA

O gerenciamento da execução das aplicações pervasivas é o objetivo do *middleware* EXEHDA, o qual é formado por diversos serviços de modo a suportar a programação e a execução de aplicações pervasivas conscientes de contexto. A sua infra-estrutura permite que as aplicações tenham a semântica *follow-me*, isto é, as aplicações do usuário podem segui-lo adaptando-se ao seu novo contexto (AUGUSTIN, 2005).

2.3.1 Reconhecimento de contexto no EXEHDA

Na arquitetura EXEHDA, o suporte à adaptação é realizado pelo subsistema de Reconhecimento de Contexto, que é utilizado como base de informações para o novo serviço de inferência de tarefas, descrito neste trabalho. Esse subsistema possui serviços responsáveis por extrair a informação bruta dos sensores e realizar as adaptações disparadas pela modificação dos elementos do contexto de interesse das aplicações (YAMIN, 2004). Os seus principais serviços são: *Collector*, *Deflector* e *ContextManager.*

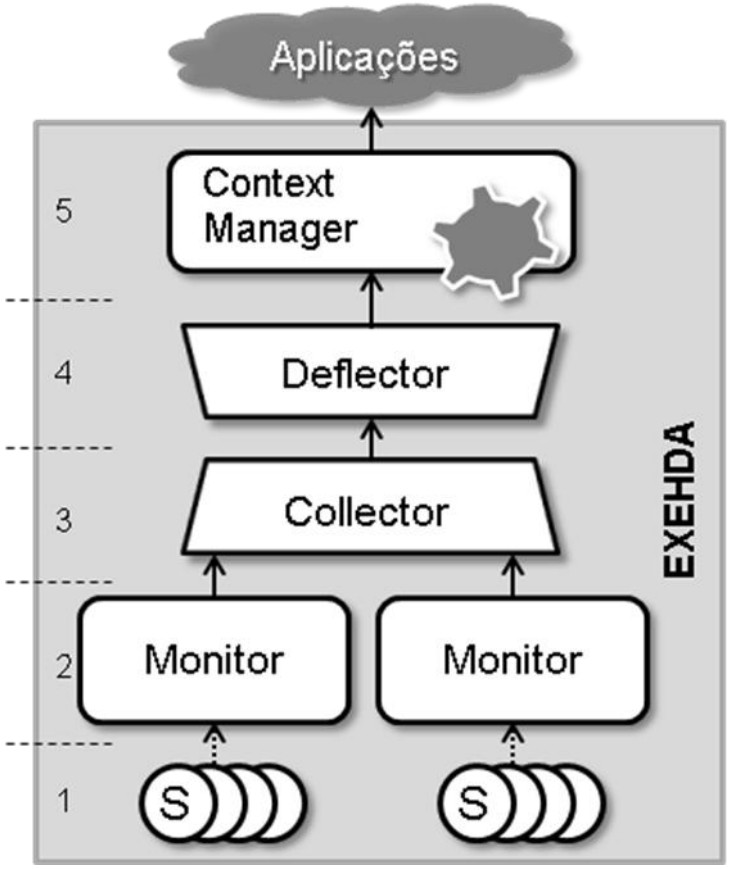

Figura 1 – Arquitetura do sistema de reconhecimento de contexto do EXEHDA

A Figura 1 ilustra em alto nível os componentes do *middleware* EXEHDA envolvidos no reconhecimento do contexto e as suas ligações entre si. O detalhamento de cada componente, bem como as suas interfaces, pode ser encontrado em (YAMIN, 2004).

No nível 1 estão os sensores que lidam diretamente com os dispositivos presentes no ambiente e repassam os dados captados para a camada 2. Caso os sensores sejam de temperatura corporal, por exemplo, o valor captado será repassado à camada superior como um dado numérico apenas.

Os monitores da camada 2 recebem os dados na forma bruta vinda dos sensores e aplicam o primeiro nível de transformação, adicionando, por exemplo, a unidade nos dados captados. No exemplo anterior, o sensor informa o número captado e o monitor adiciona o tipo de escala sendo usada, por exemplo, graus Celsius.

O nível 3, com os componentes *Collector,* é responsável por receber de tempos em tempos os dados fornecidos pelos monitores. Dentre as configurações necessárias do componente *Collector* está o intervalo de tempo de consulta aos monitores. Dessa forma, o estado do contexto é verificado eventualmente, diminuindo a necessidade de um processamento contínuo.

O nível 4 possui o componente do tipo *Deflector* que é responsável por receber as informações da camada inferior e distribuir, em modo *broadcast*, para os demais componentes interessados em receber eventos do contexto. Através dessa estratégia, unifica-se o ponto de registro para que os demais componentes tomem conhecimento de mudanças no ambiente da aplicação, além de conferir escalabilidade à arquitetura uma vez que não precisa contatar individualmente os serviços.

A última camada de reconhecimento de contexto do EXEHDA, no nível 5, possui o componente *ContextManager* que recebe as informações do componente *Deflector* e aplica as últimas transformações nos dados antes de disponibilizá-los às aplicações. Esse serviço é configurado através de arquivos XML, nos quais são definidos os tratamentos a serem feitos com os dados provenientes das camadas inferiores. Dessa forma, ocorre um pré-processamento do estado do contexto antes que ele seja disponibilizado para a aplicação, facilitando o seu desenvolvimento.

A Figura 2 demonstra um arquivo de configuração do contexto no EXEHDA. O arquivo, no formato XML, especifica os estados possíveis do contexto, indicados pela *tag* STATE, nesse caso: '*console'*, '*pda'* e '*desktop'*. Ainda, é necessário informar o nome do sensor que vai fornecer o dado para que a contexto seja reconhecido, indicado pela *tag* SENSOR, nesse caso '*runtime.gui.awt.mode'*. Por último, são definidas as regras de mapeamento entre os dados captados pelo sensor e o contexto fornecido. Na figura, nota-se que há um mapeamento indicando que, caso o sensor detecte o valor '*awt-low-res'*, será fornecido ao seu monitor o valor '*pda'*, abstraindo a informação bruta coletada no sensor.

```
−<CONTEXT n="dispositivo">
  −<STATES>
     <STATE n="console"/>
     <STATE n="pda"/>
     <STATE n="desktop"/>
   </STATES>
  −<FILTER syntax="xml:StringMapping">
   −<INDEX>
      <SENSOR n="runtime.gui.awt.mode"/>
    </INDEX>
   −<MAPPINGS>
      <MATCH op="equals" v="none" ignoreCase="true" state="console"/>
      <MATCH op="equals" v="awt-low res" ignoreCase="true" state="pda"/>
      <MATCH op="prefix" v="swing" ignoreCase="true" state="desktop"/>
      <DEFAULT state="console"/>
    </MAPPINGS>
  </FILTER>
</CONTEXT>
```

Figura 2 – Configuração do contexto no EXEHDA

Após a transformação dos dados do ambiente, as aplicações conectadas ao EXEHDA podem realizar o devido processamento para modificar o seu comportamento ou realizar as mais diversas tarefas, de acordo com as suas finalidades. O componente *ContextManager* possui dois papéis importantes para a realização do presente trabalho: é utilizado para a captura e armazenamento do estado do contexto no momento de execução de uma tarefa e; para a inferência das futuras tarefas do usuário baseado no estado atual do contexto.

3 ARQUITETURA CLINICSPACE

O capítulo apresenta a arquitetura ClinicSpace juntamente com os seus componentes. De forma a integrar a arquitetura foram desenvolvidos diversos componentes a fim de os objetivos do presente trabalho serem alcançados.

3.1 Integração ClinicSpace e EXEHDA

A arquitetura para Computação Pervasiva implementada pelo *middleware* EXEHDA é utilizada como suporte à arquitetura ClinicSpace, possibilitando o reuso dos seus componentes, dessa forma, pode-se focar nos aspectos relativos a computação orientada a atividades (*task-driven computing*) e a centralização no usuário-final (FERREIRA, 2009).

Os principais serviços do EXEHDA utilizados pela arquitetura ClinicSpace são o Gerenciamento de Contexto (*Context Manager*), o Ambiente Virtual do Usuário (AVU) e a Base de Dados Pervasiva (BDA).

O Gerenciamento de Contexto (*Context Manager*) tem grande importância para o ClinicSpace, pois é ele que controla o estado dos elementos de interesse do contexto. A partir de tal serviço, as aplicações podem registrar os seus contextos de interesse e receberem as notificações de mudança do ambiente, possibilitando o desenvolvimento de aplicações com funcionalidades conscientes de contexto.

O Ambiente Virtual do Usuário armazena as tarefas de cada usuário, construídas na Interface de Edição de Tarefas e Contexto (IETC) do ClinicSpace, detalhada na seção 3.2.2. Dessa forma, as tarefas podem ser modificadas pelos seus proprietários e tornam-se disponíveis para acesso pervasivo.

Como componente de armazenamento, é utilizada a Base de Dados Pervasiva (BDA), que disponibiliza o seu acesso de inúmeras localizações e formatos diferentes. Entre as suas responsabilidades, estão o armazenamento das definições de tarefas comuns a todos os usuários do sistema, bem como o próprio código-fonte de algumas funcionalidades necessárias ao ClinicSpace.

A seguir, são detalhados os componentes que formam a arquitetura ClinicSpace.

3.2 Componentes da Arquitetura ClinicSpace

A arquitetura ClinicSpace foi desenvolvida para se tornar um ambiente voltado às atividades clínicas em conjunto com um Sistema de Informação em Saúde (pEHS – *pervasive Electronic HealthCare System*) (VICENTINI, 2010) e o ambiente de execução pervasivo, gerenciado pelo *middleware* EXEHDA (YAMIN, 2004). A arquitetura, demonstrada na Figura 3, possui componentes bem definidos que permitem a modelagem de tarefas pelo usuário-final e o respectivo gerenciamento do seu ciclo de vida. A seguir, é apresentada de uma forma geral a arquitetura, porém os detalhes de cada desenvolvimento podem ser encontrados nas referências (FERREIRA, 2009a), (MACHADO, 2010), (RIZETTI, 2009), (SILVA, 2009a) e (VICENTINI, 2010)

Na Figura 3, os serviços resultantes do trabalho de desenvolvimento dessa dissertação são apresentados em destaque (cinza escuro) e serão detalhados na seção 4.3.

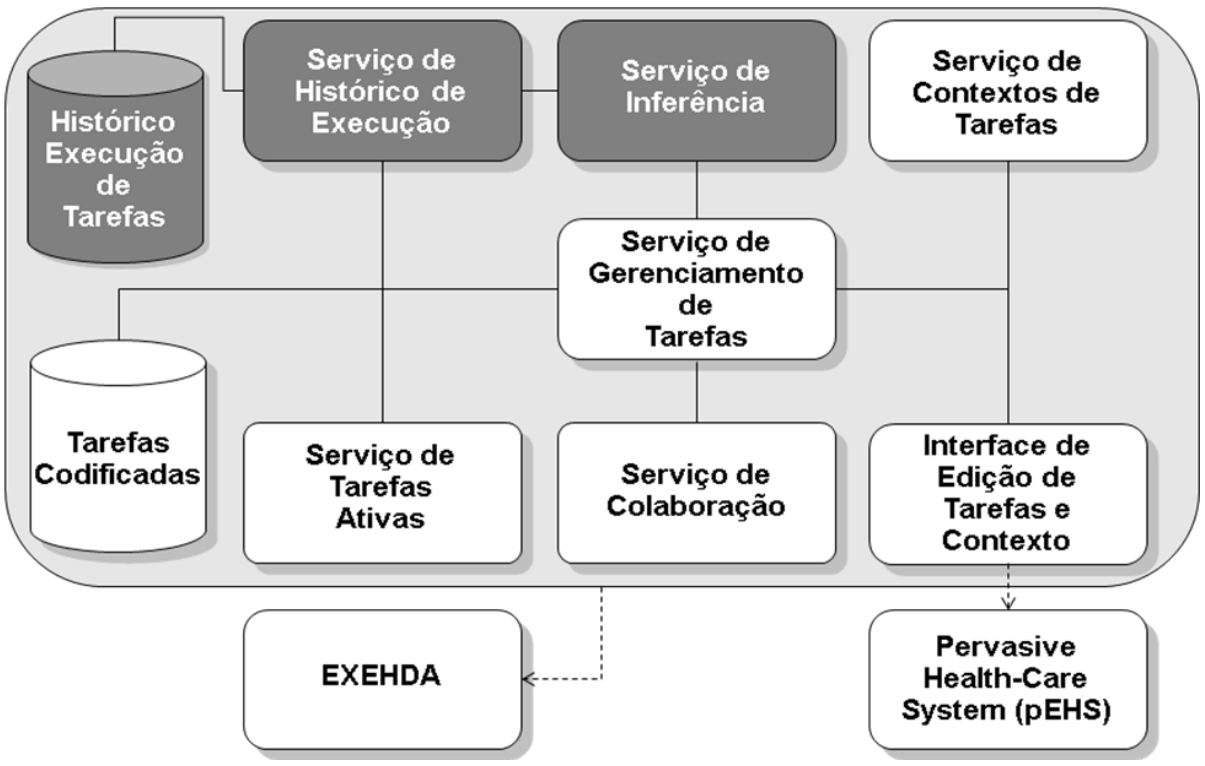

Figura 3 – Componentes da Arquitetura do projeto ClinicSpace

3.2.1 pEHS – Sistema de Informação Pervasivo em Saúde (*pervasive Eletronic HealthCare System*)

De maneira a permitir a utilização e validação da arquitetura ClinicSpace, foram pesquisados Sistemas Eletrônicos de Saúde (EHS) existentes e, a partir desse estudo, foi projetado um novo sistema com características de pervasividade, chamado pEHS (VICENTINI, 2010). Entre as principais características dessa aplicação projetada estão: colaboração entre profissionais para a realização de tarefas, compartilhamento de informações e suporte à mobilidade.

As funcionalidades projetadas para o pEHS foram desenvolvidas na menor granularidade possível para que elas possam ser utilizadas isoladamente para os mais diversos fins na composição das tarefas dos usuários. Com isso, adotou-se o conceito de subtarefa que representa o mapeamento entre uma funcionalidade presente no pEHS e o restante da arquitetura ClinicSpace. Como exemplo de subtarefa, pode-se citar: “identificação de paciente”, “busca de exames visuais do paciente”, etc.

A composição de várias subtarefas compõe a tarefa a ser executada pelo usuário, representando o fluxo de execução necessário para a realização de uma ou várias atividades do seu cotidiano.

Fica evidente, com isso que o pEHS deve ser altamente modularizado, permitindo que cada funcionalidade possa ser mapeada para uma subtarefa e estas possam formar as atividades. Caso o pEHS possua uma granularidade muito alta das suas funções, ou seja, agrupando várias funcionalidades que não podem ser decompostas, corre-se o risco de que as tarefas modeladas pelo usuário não representem a melhor forma de realização do seu trabalho (VICENTINI; MACHADO; AUGUSTIN, 2009). Essa inflexibilidade do sistema poderia levar a uma rejeição do sistema pelos usuários, contrariando o objetivo principal da arquitetura ClinicSpace que é permitir que o usuário modele o sistema de acordo com as suas rotinas (FERREIRA et al, 2009b).

O detalhamento do desenvolvimento e demais requisitos necessários a um pEHS podem ser encontrados em (VICENTINI, 2010).

3.2.2 Interface de Edição de Tarefas e Contexto (IETC)

Com o intuito de disponibilizar ao usuário uma ferramenta de modelagem de suas

tarefas, está sendo desenvolvida a Interface de Edição de Tarefas e Contexto. Essa ferramenta-piloto é constituída de uma interface para que o usuário modele/programe as suas tarefas de acordo com a sua forma particular/personalizada de realizar as atividades clinicas diárias.

Para compor uma determinada tarefa, o usuário seleciona as subtarefas disponíveis, relacionadas às funcionalidades mínimas disponíveis no pEHS, e as ordena de maneira a refletir o seu fluxo de trabalho Dessa forma, a execução das tarefas no sistema ocorrerá de acordo com o modo de trabalho de cada usuário, garantindo a personalização do sistema (SILVA, 2009b).

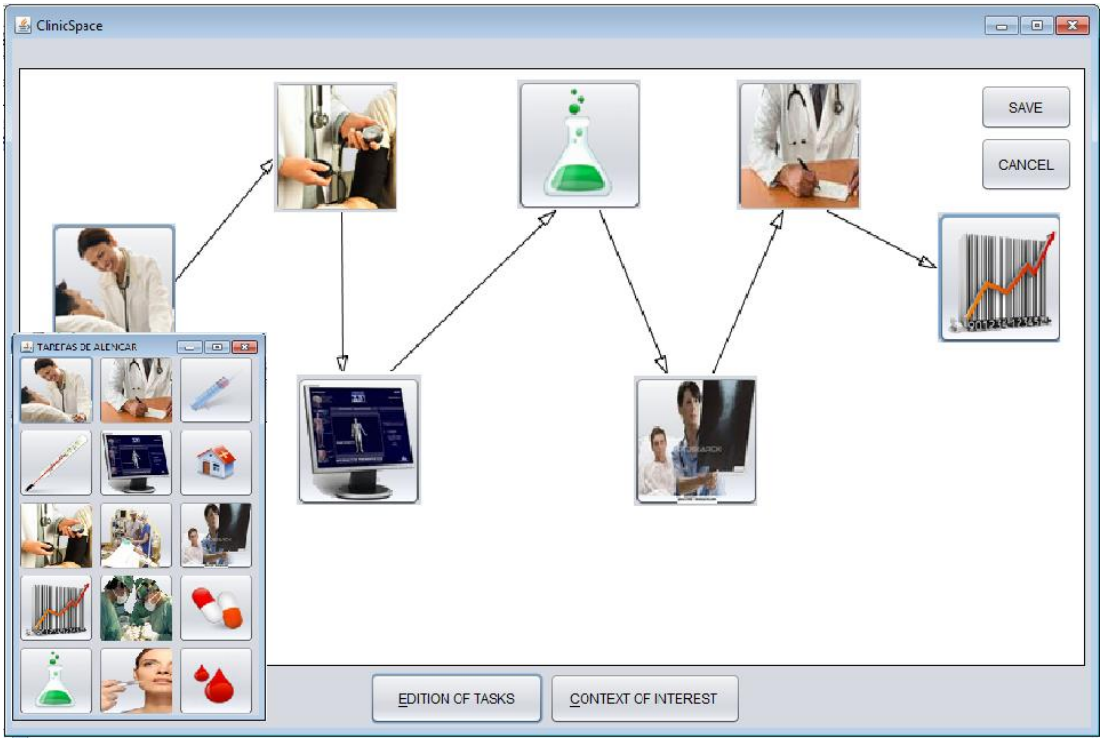

Figura 4 – Interface de Edição de Tarefas e Contexto (MACHADO, 2010)

A Figura 4 ilustra a interface gráfica da Interface de Edição de Tarefas e Contexto (IETC). Pode-se perceber a presença de uma janela que disponibiliza ao usuário as subtarefas passíveis de serem usadas, representadas por ícones de forma a tornar seu uso mais natural. Conforme a tarefa modelada, o usuário escolhe as subtarefas desejadas e as arrasta para o painel, aonde deve ser especificada a ordem, através de setas, na qual a execução deve ocorrer, formando o seu *workflow*.

Como forma de implantar características do ambiente nas tarefas, o usuário tem a opção de anexar elementos do contexto disponível para cada subtarefa, ao acionar o botão "*Context of Interest*" (MACHADO, 2010). A interface, representada pela Figura 5, exibe

quais os elementos do contexto podem ser introduzidos em cada subtarefa, bem como um formulário no qual o usuário especifica o comportamento do sistema em relação ao estado do contexto.

A partir dessas configurações, o usuário pode configurar quais alertas serão emitidos quando uma determinada condição for satisfeita, permitindo que o sistema aja pró-ativamente a seu favor. Assim, as características do contexto influenciam no comportamento da aplicação e podem auxiliar o usuário na tomada de decisões, por exemplo.

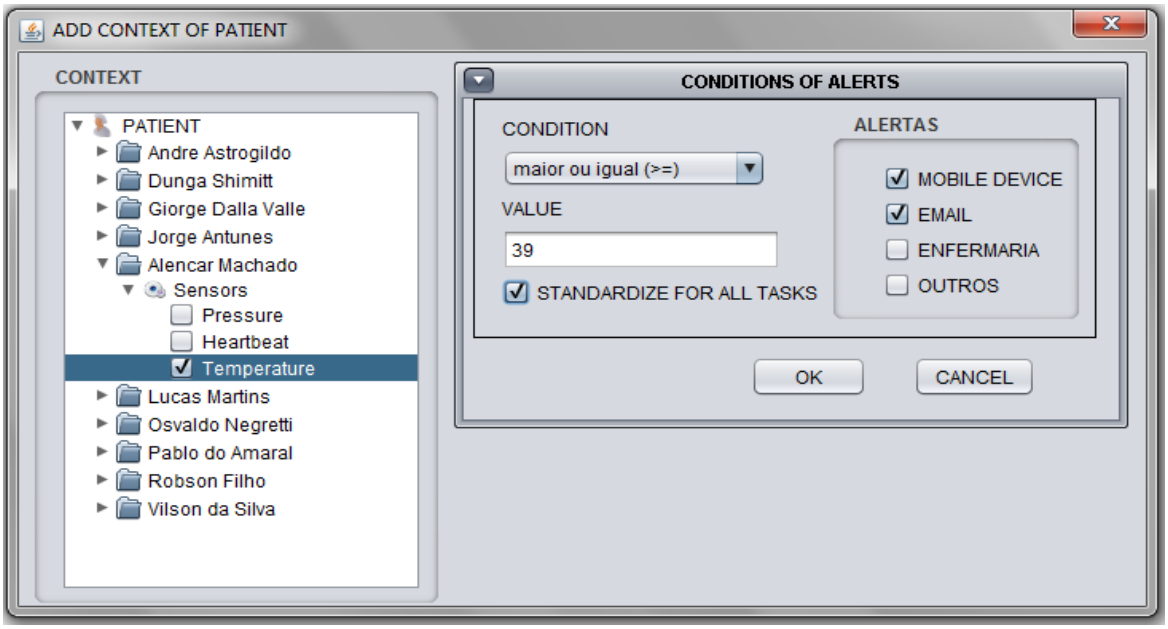

Figura 5 – Associação do contexto com subtarefas (MACHADO, 2010)

Após finalizar a modelagem da tarefa, o usuário salva a modelagem/programação realizada e o sistema a armazena como um descritor XML, contendo todas subtarefas envolvidas, bem como a seqüência configurada. No momento de execução de uma tarefa, o Sistema de Gerenciamento de Tarefas (SGT) identifica as subtarefas que compõem a tarefa e as instancia conforme a sua execução acontece, obedecendo às configurações realizadas na sua modelagem feita na IETC.

A associação dos elementos do contexto com as subtarefas, realizada pelo próprio usuário é o trabalho em desenvolvimento da dissertação do mestrando Alencar Machado[1]. Além disso, essa interface terá funcionalidades para que o usuário, além de modelar as suas tarefas, possa também executá-las tornando-se o ponto único de utilização do sistema.

[1] Integrante do GMob e mestrando do Programa de Pós-Graduação em Informática da Universidade Federal de Santa Maria.

3.2.3 Sistema de Contextos de Tarefas (SCT)

Esse serviço armazena o relacionamento das tarefas do usuário com o contexto configurado no momento da modelagem. Além disso, o SCT identifica elementos do contexto que podem servir para a diminuição da entrada manual de dados nos formulários do pEHS, facilitando o seu uso e diminuindo a propensão a erros humanos (RIZETTI, 2009). Ainda, o usuário pode realizar o agendamento de execução de determinadas tarefas quando o contexto estiver em um estado pré-determinado. O desenvolvimento desse sistema é tratado em detalhes no trabalho realizado por Rizetti (RIZETTI, 2009).

3.2.4 Sistema de Gerenciamento de Tarefas (SGT)

A funcionalidade desse sistema é fornecer um mapeamento entre as subtarefas do ClinicSpace e os serviços da plataforma EXEHDA, usada para prover a pervasividade do sistema como um todo. Em alguns casos, uma subtarefa pode envolver a execução de diversos serviços EXEHDA para que a sua funcionalidade seja alcançada.

O principal papel desse serviço é tratar o mapeamento entre as funcionalidades do sistema pEHS e as subtarefas modeladas nas tarefas do usuário. Assim, sempre que uma subtarefa for executada, esse serviço identifica qual a respectiva funcionalidade do sistema pEHS deve entrar em execução e invoca os serviços EXEHDA. É responsabilidade do SGT, ainda, gerenciar o ciclo de vida das tarefas. Caso o usuário necessite parar temporariamente ou finalizar uma tarefa, o SGT executa as operações necessárias para atualizar o seu estado ou terminá-la, mantendo assim o controle de execução do sistema (FERREIRA, 2009a).

O SGT utiliza a base de Tarefas Codificadas que contém os descritores das tarefas modeladas de cada usuário, armazenadas na forma de arquivos XML. No momento em que uma tarefa é iniciada, o SGT busca o seu descritor XML e identifica a seqüência de subtarefas necessárias. De acordo com o andamento da execução, o sistema instancia as demais subtarefas necessárias até que a tarefa seja concluída.

Esse serviço atua como servidor para a maioria dos demais componentes da arquitetura, uma vez que é o responsável por instanciar e executar as tarefas sob demanda do usuário ou do próprio sistema.

3.2.5 Sistema de Tarefas Ativas (STA)

Esse serviço tem como objetivo conhecer quais as tarefas estão correntemente em

execução pelos usuários do sistema para realizar a migração entre dispositivos de acordo com o deslocamento do usuário. Para tal, o STA utiliza serviços do *middleware* EXEHDA, para identificar a movimentação do usuário e realizar a semântica *follow-me*, trazendo dinamicidade ao sistema como um todo (FERREIRA, 2009a).

Devido a sua natureza, possui uma forte ligação com o SGT, citado anteriormente, uma vez que é informado sempre que uma tarefa entre em execução ou é finalizada.

3.2.6 Serviço de Colaboração

O Serviço de Colaboração tem como objetivo permitir a execução compartilhada das tarefas dos usuários. A partir dessa funcionalidade, o usuário pode decidir, em uma determinada etapa da realização de uma tarefa, que algum outro usuário o auxilie.

Dessa forma, o sistema se torna mais próximo do ambiente clínico, aonde a tomada de decisão muitas vezes requer a colaboração de vários outros usuários com especialidades distintas (BARDRAM, 2003), (BARDRAM, CHRISTENSEN, 2007). O desenvolvimento desse serviço é responsabilidade do mestrando Marcelo Lopes Kroth[2].

3.2.7 Serviço de Histórico de Execução

Esse componente da arquitetura tem como objetivo armazenar o histórico de execução das tarefas executadas pelos usuários, assim como o contexto captado no início da execução das tarefas. A partir desse armazenamento, realizado em Banco de Dados relacional, pode-se traçar o perfil de utilização do usuário e, com isso, realizar inferências sobre as tarefas a serem executadas no futuro.

Outra vantagem da utilização do histórico é a criação de *logs* da utilização do sistema permitindo auditorias sobre as ações tomadas pelos usuários, bem como das circunstâncias (contexto) presentes na execução das tarefas. Esse componente será abordado em mais detalhes na seção 4.3.2.

3.2.8 Serviço de Inferência ou Serviço de Predição de Tarefas (SPT)

Esse serviço tem a responsabilidade de inferir as próximas tarefas do usuário de acordo com o seu histórico de execução e a situação do ambiente captada no momento da sua execução. Para a realização desse objetivo, o Serviço de Inferência possui uma forte interação

[2] Integrante do GMob e mestrando do Programa de Pós-Graduação em Informática da Universidade Federal de Santa Maria.

com os serviços citados anteriormente (SGT e SHE), assim como com os componentes do *middleware* EXEHDA para a captura do contexto.

De forma a não tornar os serviços envolvidos na inferência de tarefas demasiadamente acoplados entre si, utilizou-se uma arquitetura com interfaces bem definidas e adoção de padrões de projeto, visando a facilitação de manutenções futuras.

Para a inferência das tarefas em si, o sistema utiliza o algoritmo C4.5 (QUINLAN, 1993) para a construção de uma árvore de decisão sobre os elementos do contexto. Os detalhes dessa implementação são demonstrados nas seções 4.3.3 e 4.3.4.

4 SERVIÇO DE INFERÊNCIA DE TAREFAS

Este capítulo descreve o serviço de inferência de tarefas desenvolvido no presente trabalho que complementa a arquitetura ClinicSpace.

4.1 Requisitos da Inferência de Tarefas

A inferência de tarefas pode ser considerada uma forma de pró-atividade do sistema em relação ao usuário, sendo um dos requisitos desejáveis de um sistema pervasivo. A partir da observação de determinados fatores, o *software* decide tomar alguma decisão e prover uma funcionalidade específica ao usuário sem que o mesmo tenha atuado explicitamente. Isto contribui para tornar a computação mais transparente para o usuário-final.

Pode-se utilizar a inferência como forma de prevenir certos acontecimentos durante a utilização de um sistema ou dispositivo. O sistema sendo acessado pelo usuário através de um PDA, por exemplo, pode detectar que a sua localização não possui uma boa conectividade de rede e avisá-lo sobre tais condições. Ainda, o sistema pode decidir desabilitar determinadas funcionalidades até que as condições normais sejam restabelecidas.

Para a realização da pró-atividade dos sistemas, normalmente, são usados agentes autônomos de *software* (MAYHROFER, 2004). Tais componentes são espalhados pela arquitetura em questão e possuem certa inteligência para reagir a determinados estímulos e auxiliar o usuário nas suas atividades.

Uma maneira alternativa à utilização de agentes de *software* pode ser o monitoramento dos elementos de um contexto de interesse através do uso de sensores. A inferência de tarefas baseada em sensores permite que haja uma resposta do sistema, de forma proativa, baseada na reação às mudanças dos elementos de contexto. A Figura 6 exemplifica de que forma as informações podem ser extraídas do ambiente com o uso de sensores e o envolvimento do sistema para o tratamento dessas informações.

Os sensores monitoram as mudanças do ambiente e as repassam para o sistema, que se torna consciente de contexto. A partir disso, a aplicação pode realizar diversas operações,

como modificar a interface do usuário a fim de melhorar a usabilidade do sistema.

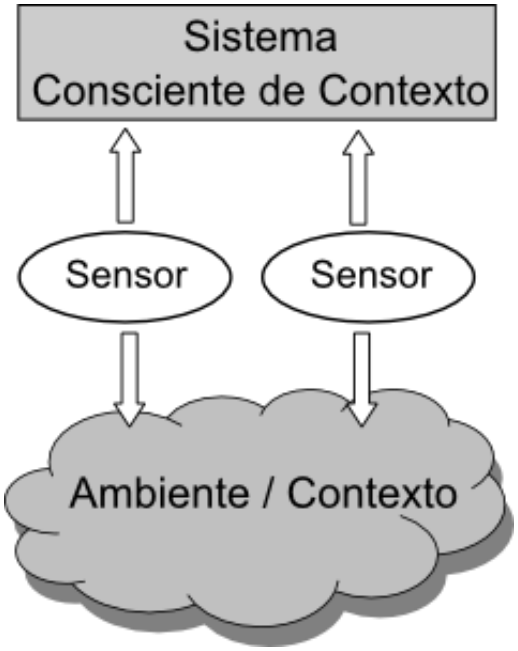

Figura 6 – Sistema consciente de contexto

As informações captadas pelos sensores e repassadas ao sistema podem servir, entre outras coisas, para a realização de predição/inferência de tarefas, no entanto, é necessário que exista uma arquitetura de componentes bem definida para a sua realização. Dessa forma, a implantação de inferência de tarefas em um sistema já existente torna-se viável, pois as responsabilidades de cada componente estão definidas visando o desacoplamento e isolamento. Caso sejam necessárias mudanças internas aos componentes, os seus clientes não são afetados, o que facilita a manutenção geral da arquitetura.

4.2 Arquitetura Padrão para Inferência

A Figura 7 exibe a definição padrão de uma arquitetura para a predição do próprio contexto (MAYHROFER, 2004), porém essa mesma arquitetura pode ser adaptada para a predição de tarefas. Na figura, no nível 1, os sensores monitoram o contexto e extraem as informações brutas do ambiente. Nesse nível, um sensor de temperatura, por exemplo, forneceria apenas o número captado do ambiente. Na arquitetura EXEHDA, essa camada é composta pelos sensores dos elementos de contexto.

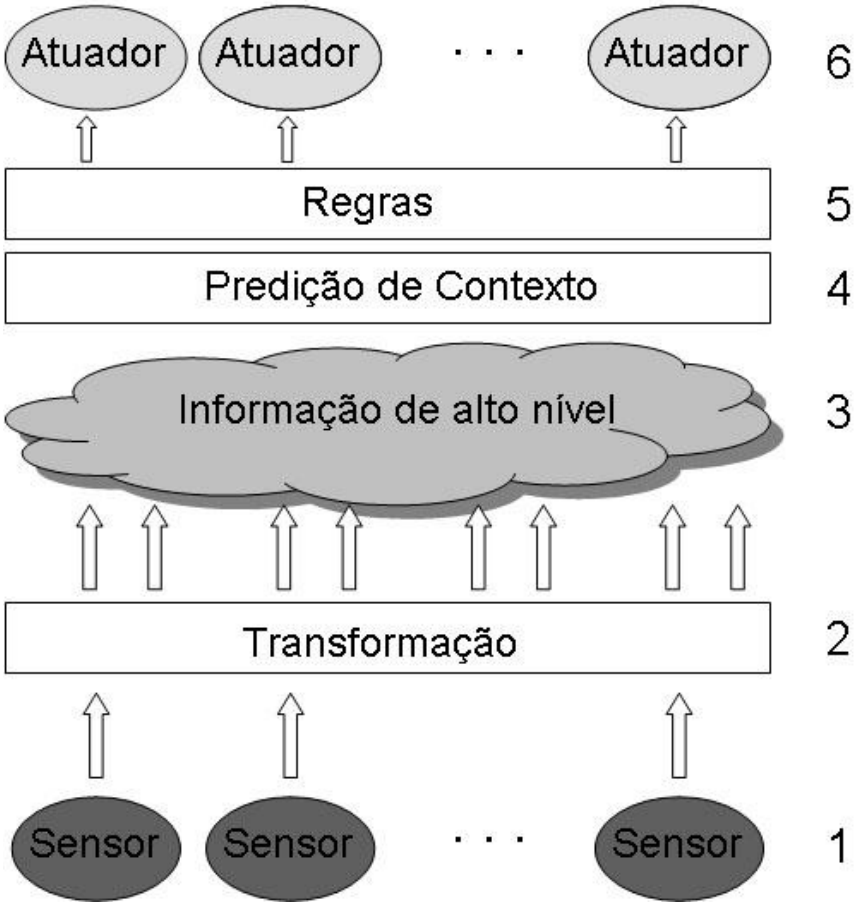

Figura 7 – Arquitetura, em alto nível, de um sistema de predição de contexto (MAYHROFER, 2004).

A camada do nível 2 é responsável por transformar a informação bruta captada pelo sensor. No exemplo anterior, após a transformação aplicada nessa camada, o resultado seria a temperatura juntamente com a sua unidade (graus Celsius ou Fahrenheit) a ser usada pela camada superior.

A próxima camada, do nível 3, interpreta a informação recebida e, de acordo com regras, disponibiliza a informação em um nível mais abstrato. No exemplo usado, essa camada recebe a temperatura captada e fornece um valor indicando se a mesma é baixa, média ou alta. Essa camada necessita de uma prévia configuração para que essa interpretação ocorra e o resultado esperado seja disponibilizado. No *middleware* EXEHDA, os monitores recebem os dados dos sensores e, de acordo com regras configuradas em arquivos XML, disponibilizam informações em alto nível para as camadas superiores, conforme resumido na seção 2.3.1.

Após a extração em alto nível das informações do contexto, Mayhrofer (MAYHROFER, 2004) exibe um componente no qual é feita a predição de elementos (nível 4 da Figura 7) de contexto. Nesse caso, a arquitetura pode ser usada para prever o estado de

alguns elementos de contexto, no caso de eles não estarem disponíveis no momento. Para a predição de tarefas, realizada neste trabalho, essa camada é responsável por usar as informações de contexto disponíveis e prever a próxima tarefa a ser executada pelo usuário. Para isso, é necessário o auxílio do histórico de execução de tarefas de cada usuário.

O nível 5 possui as regras, baseadas nas informações processadas pelas camadas inferiores, para que a camada superior (nível 6) de atuadores seja ativada. A camada de atuadores tem como função intervir no sistema, de forma proativa, seja provendo dados para o usuário, seja auxiliando em alguma funcionalidade. No caso da predição de tarefas, essa camada é responsável por apresentar as sugestões de tarefas a serem executadas pelo usuário, de forma não intrusiva.

A arquitetura sugerida por Mayhrofer (MAYHROFER, 2004) foi utilizada como referência no desenvolvimento dos componentes da arquitetura ClinicSpace para a realização do serviço de inferência de tarefas. Dentre os principais benefícios dessa utilização está a modularidade e a divisão explícita das responsabilidades das camadas, facilitando o desenvolvimento e permitindo que os detalhes de implementação de cada camada sejam abstraídos pelas demais.

4.3 Arquitetura do Sistema de Inferência de Tarefas

Nessa seção, são apresentados os serviços desenvolvidos nesse trabalho para a realização da inferência de tarefas, bem como as interações com o serviços já previamente existentes na arquitetura ClinicSpace e com o *middleware* EXEHDA. Primeiramente, são detalhados os principais serviços envolvidos (Serviço de Contexto para o Histórico de Execução, Serviço de Histórico de Execução e Serviço de Predição de Tarefas) e após é descrito o serviço desenvolvido para auxiliar na arquitetura de inferência: Serviço Spring.

4.3.1 Serviço de Contexto para o Histórico de Execução (*ExecutionHistoryContextService*) (SCHE)

O Serviço de Contexto para o Histórico de Execução (SCHE) tem por objetivo definir

quais aspectos do contexto são de interesse para armazenar no histórico de execução. Esse serviço especifica quais sensores e monitores são consultados durante a execução de uma tarefa, de modo a retratar o momento da execução das tarefas.

Qualquer serviço do ClinicSpace interessado em receber as notificações de modificação dos elementos do contexto pode se inscrever no SCHE. Dessa forma, as notificações geradas podem ser usadas para diversas funcionalidades e diferentes serviços, tornando a arquitetura flexível.

Para que a captura do contexto seja feita, é utilizado o componente *ContextManager* do *middleware* EXEHDA. Durante a sua instanciação, o SCHE se inscreve no *ContextManager* para receber as mudanças dos elementos de contexto de interesse do serviço. Dessa forma, apenas os elementos que importam para a SCHE são consultados.

No *ContextManager*, o SCHE registra-se para receber as alterações do sensor *Device*, já presente no *middleware* EXEHDA, e dos sensores desenvolvidos especialmente para complementar o presente trabalho: *Sector*, *Time Frame*, *People Around* e *Devices Around*. O sensor *Device* monitora o dispositivo principal na qual o usuário acessa o sistema pEHS. Essa informação permite que se saiba a preferência de dispositivo do usuário para executar suas tarefas. Os possíveis valores retornados são:

- *PDA (Personal Digital Assistant)*: indica a utilização de um dispositivo móvel ao acessar o sistema;
- *Console*: indica que o sistema está sendo utilizado através de um console de texto;
- *Desktop*: indica a utilização de um computador de mesa.

O sensor *Sector* foi desenvolvido nesse trabalho para detectar a localização física do usuário na forma de setores de uma unidade, como um hospital, espaço pervasivo foco do projeto ClinicSpace. Tal informação permite o armazenamento do local de preferência para a realização de cada tarefa do pEHS. Os possíveis valores retornados são:

- Corredor: indica que o usuário não está em nenhuma área específica e, sim, em deslocamento para algum setor específico;
- Emergência: indica a utilização do sistema no setor de emergências do hospital;
- Cirurgia: indica a utilização do sistema no setor de cirurgias;
- Clínica: valor retornado quando o usuário está em sua clínica prestando atendimento a um paciente;

- Enfermaria: valor que representa que o usuário está na enfermaria, durante a execução do sistema. Isso pode acontecer, por exemplo, no momento da conferência de medicações.
- Desconhecido: significa que o usuário está presente em alguma localização não cadastrada no sistema, diferente das demais citadas acima.

Ao contrário de se detectar a hora exata da execução de uma tarefa, escolheu-se usar o conceito de *time frames*, medido em minutos, monitorado pelo sensor *Time Frame*, desenvolvido durante a realização dessa dissertação. Dessa maneira, é armazenado, de forma relativa, o momento de execução de uma tarefa, melhorando a precisão da inferência de tarefas. Essa abordagem permite que o sistema trace o perfil de utilização dos usuários, possibilitando que se saiba que o usuário executa uma determinada tarefa no mesmo período de tempo todos os dias.

Adotou-se o tamanho de cada *time frame* de 10 minutos devido a ser este o tempo médio necessário para a realização das tarefas mais comuns no cotidiano clínico (LAERUM, 2004). Dessa forma, a cada 1h do dia existem 6 *time frames* e, em um período de 24 horas são 144 *time frames*. Para a realização do cálculo, é capturada a quantidade de minutos do instante atual a partir da zero hora do dia. Após, divide-se esse resultado pelo tamanho configurado para o *time frame* adicionando-se 1. Exemplificando: caso o usuário execute a tarefa "Atendimento ao Paciente" às 0h e 01min, o *time frame* dessa execução será 1; caso uma tarefa seja executada às 8h e 12min. o seu *time frame* será 50.

A quantidade de minutos de um *time frame* pode ser especificada para o sensor através de um arquivo de configuração, sem a necessidade de alteração do código-fonte, facilitando a configuração do sistema. A utilização de *times frames* grandes agrupa diversas tarefas, o que aumenta o grau de confiança da inferência, porém, corre-se o risco de perder particularidades da execução de tarefas do usuário. No entanto, *times frames* menores diminuem a quantidade de tarefas agrupadas e prejudicam a inferência de tarefas que tenham duração maior que esse período (KALATZIS, 2008).

O sensor *People Around* foi desenvolvido com a responsabilidade de detectar as pessoas ao redor usuário no momento da execução de uma tarefa. Essa informação do contexto complementa as demais e auxilia na inferência de tarefas, pois forma o contexto de execução da tarefa. Esse sensor é capaz de detectar se o usuário está cercado de enfermeiros, médicos, pacientes, etc.

O sensor *Device Around* capta os aparelhos e dispositivos presentes no ambiente, ao redor do usuário. Essa informação torna-se importante de ser captada, pois, de acordo com o local em que se encontra, pode-se detectar, por exemplo, aparelhos de Raio-X, macas, equipamentos de Eletrocardiograma, etc, complementando as informações do contexto.

A escolha da captura dos dados de tais sensores foi devido a eles formarem o ambiente no qual o usuário está inserido, de acordo com os três aspectos: “Aonde você está” (*Where are you*), “Quem está com você” (*Who you are with*) e, “Quais recursos estão próximos a você” (*What resources are nearby*) (SCHILIT, 1994). Dessa forma, o instante e a forma na qual o usuário utilizou o sistema são armazenados, permitindo com que o seu perfil de execução seja o mais real e preciso possível.

4.3.2 Serviço de Histórico de Execução (*ExecutionHistoryService*) (SHE)

O Serviço de Histórico de Execução (SHE) é responsável por armazenar o histórico de execução das atividades dos usuários do sistema. Além de serem armazenadas as tarefas executadas, o estado dos elementos de contexto no momento da execução também é mantido.

No momento em que um usuário executa uma determinada tarefa, o Sistema de Gerenciamento de Tarefas (SGT) informa o SHE, que por sua vez captura o estado do contexto utilizando o SCHE. As informações do contexto, a tarefa em questão e o seu momento de início, são armazenados no banco de dados formando o perfil de execução de tarefas do usuário.

A finalização de uma tarefa por parte do usuário faz com que o SGT atualize o histórico da tarefa, informando o SHE. Dessa forma, o início da tarefa e o estado do contexto presente nesse momento são mantidos, bem como o seu instante de término.

O armazenamento dos dados utiliza o *framework* de Mapeamento Objeto-Relacional (*ORM – Object Relational Mapping*) Hibernate (BAUER; KING, 2007). Tal ferramenta permite que as informações sejam manipuladas na forma de objetos e isola as características específicas dos Sistemas Gerenciadores de Bancos de Dados (SGBD). Essa estratégia permite ao desenvolvedor que, uma vez configurado tal mapeamento, o sistema trabalhe apenas com objetos e sejam abstraídas informações como nomes de colunas, tabelas, etc, facilitando a manutenção do sistema.

O mapeamento dos objetos para as tabelas do banco de dados é feito através de anotações[3] no código-fonte do sistema, facilitando a sua manutenção e desenvolvimento, uma vez que as configurações são facilmente percebidas. A Figura 8 mostra o mapeamento da classe `Task` para a tabela `EXEHDA_TASK` do banco de dados. A anotação `@Entity` especifica que o objeto representa uma entidade com características de persistência. Pode-se perceber, ainda, a anotação `@Column`, que indica o mapeamento da propriedade `id` para a coluna `id` da tabela EXEHDA_TASK. O mesmo acontece para a propriedade `name`.

```
@Entity
@Table(name="EXEHDA_TASK")
public class Task {
    @Id
    @Column(name="id")
    private Long id;
    @Column(name="name")
    private String name;
```

Figura 8 – Mapeamento Objeto-Relacional

As informações sobre o mapeamento dos objetos para as tabelas e suas respectivas propriedades para as colunas são lidas no momento da construção do SHE. Dessa forma, sempre que for necessário manipular (salvar, atualizar, remover) uma entidade, o sistema já tem conhecimento sobre quais tabelas deve operar. A arquitetura construída para a utilização do *framework* Hibernate tornou mais prática essa mesma adoção por futuros serviços a serem desenvolvidos. Caso novos sistemas necessitem persistência de dados, poderão reutilizar o serviço já desenvolvido.

4.3.3 Serviço de Predição de Tarefas (SPT)

O Serviço de Predição de Tarefas (SPT - *PredictionService*) tem a responsabilidade de inferir as futuras tarefas dos usuários baseado no histórico de execução de tarefas, utilizando o SHE. A partir da construção de uma árvore de decisão com os dados do histórico de execução

[3] Recurso para a declaração de meta dados e configurações, de forma declarativa, em classes Java (SUN MICROSYSTEMS, 2008b).

do usuário, o SPT monitora o ambiente e, sempre que infere a execução de uma determinada tarefa, notifica a interface gráfica (Interface de Edição de Tarefas e Contexto) para que essa sugira a próxima tarefa.

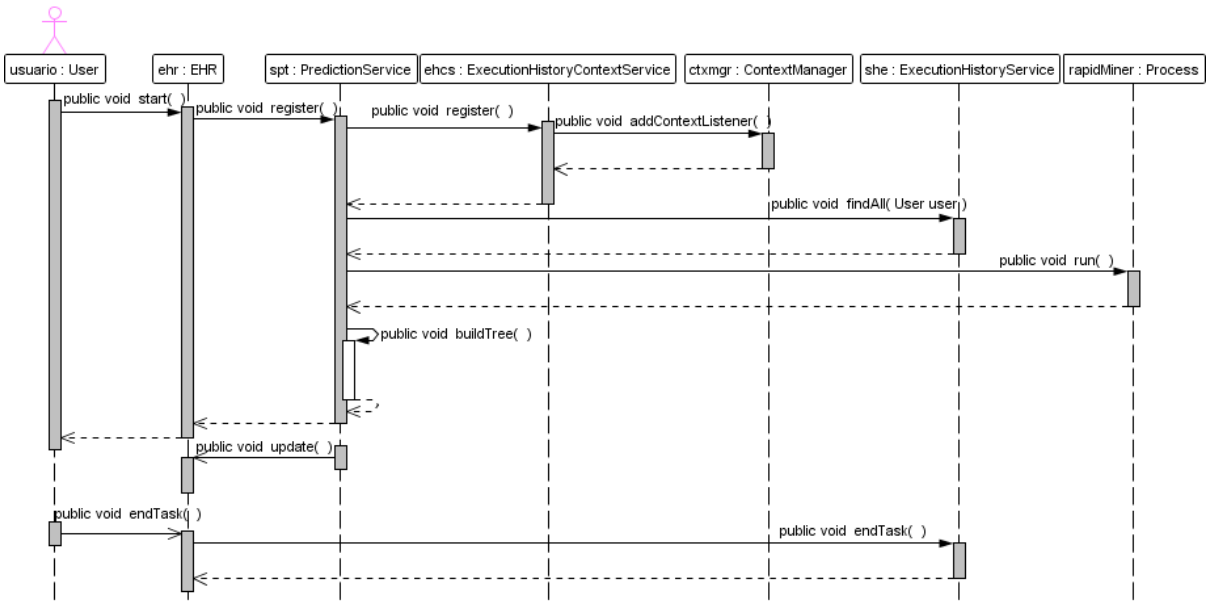

Figura 9 – Diagrama de sequência do funcionamento da inferência de tarefas

A Figura 9 exibe, de forma ampla, o diagrama de seqüência do funcionamento do sistema a partir do momento em que o usuário inicializa a interface (IETC) até a notificação de uma tarefa inferida. O usuário inicia a execução da IETC a fim de realizar as suas tarefas cotidianas, esse último por sua vez, registra-se no serviço *PredictionService* (SPT) para o recebimento de eventos de inferência. Dessa forma, sempre que for detectada a inferência de uma tarefa, a IETC toma conhecimento e pode sugeri-la ao usuário.

Logo após, o serviço *PredictionService* se registra no serviço *ExecutionHistoryContextService* (SCHE) para receber os eventos do contexto de interesse do sistema. Esse serviço, por sua vez, registra-se no componente *ContextManager* do *middleware* EXEHDA para o recebimento dos eventos de modificação dos sensores utilizados no sistema (*Time Frame, Device, Sector, People Around e Devices Around*).

O próximo passo é o serviço *PredictionService* realizar a busca dos históricos de execução do usuário corrente do sistema para invocar o componente *Process* (método *run*) da API do projeto RapidMiner (MIERSWA, 2006). Esse componente é responsável por aplicar o algoritmo C4.5 (QUINLAN, 1993) e construir a árvore de decisão que servirá como base para a inferência de tarefas.

Após a execução do componente *Process*, o resultado é processado de forma a ser construída uma Árvore de Decisão (*Decision Tree*) mais adequada para a consulta. Sempre que o SCHE detectar mudanças no contexto, o SPT deve realizar uma navegação na Árvore de Decisão a fim de verificar se uma tarefa pode ser inferida ou não. Caso seja possível, o SPT invoca a IETC (método *update*) enviando o nome da tarefa inferida.

Após o recebimento da tarefa inferida, a interface exibe-a como sugestão ao usuário da próxima tarefa a entrar em execução. O usuário, por sua vez, tem a opção de iniciar essa tarefa ou simplesmente ignorá-la prosseguindo com o seu trabalho normalmente.

No momento em que o usuário termina uma tarefa, a interface notifica o SHE para que esse armazene o instante de término da tarefa.

Devido a necessidade de utilização de métodos assíncronos entre os serviços (atualização do contexto, detecção de tarefa inferida, etc.), utilizou-se o padrão de projeto *Listener*. Esse padrão permite que um determinado componente registre-se em outro para o recebimento de eventos de seu interesse, de maneira análoga a um ouvinte (GAMMA, et al, 1995). Dessa forma, os serviços não precisam conhecer os detalhes de implementação entre eles, bastando manter interfaces de métodos bem definidas.

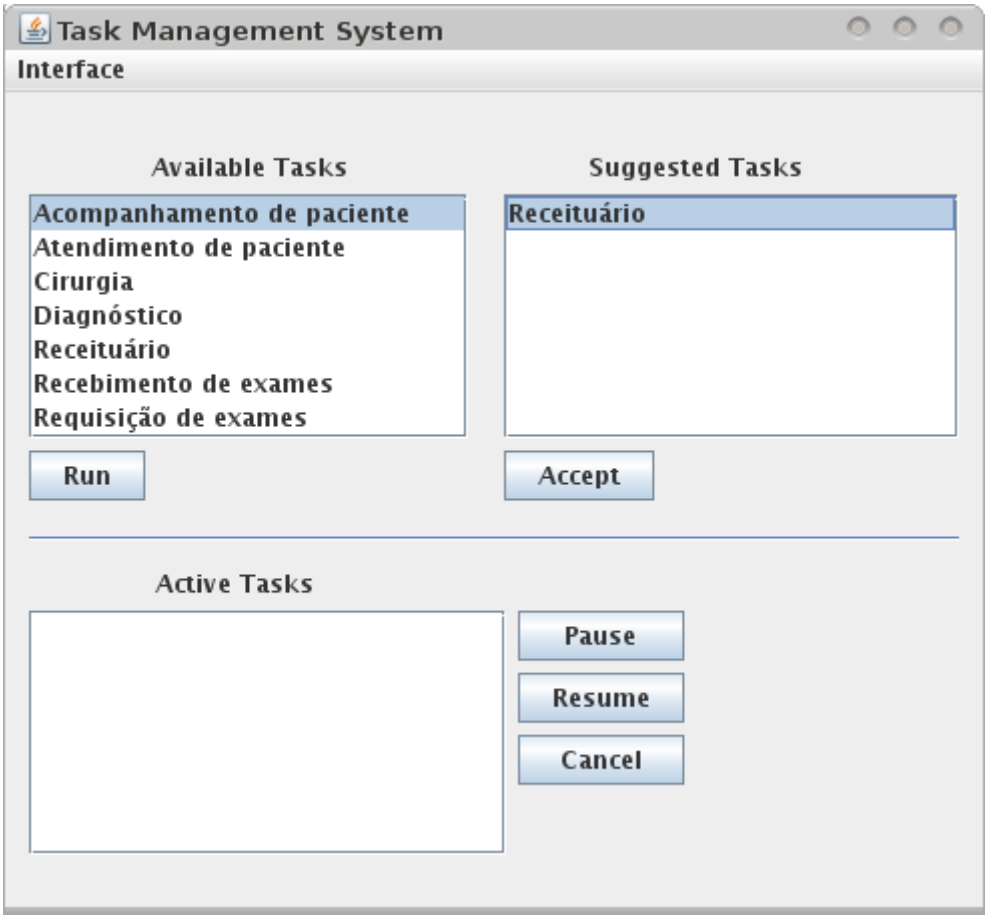

Figura 10 – Interface de execução de tarefas

Como a Interface de Edição de Tarefas e Contexto está sendo desenvolvida atualmente[4], foi simulada a execução das tarefas na interface ilustrada na Figura 10. Nela é possível notar, na parte esquerda superior, as tarefas que foram configuradas pelo usuário. Sempre que desejar, o usuário pode selecionar a tarefa de seu interesse e clicar no botão "*Run*" para que a sua execução seja iniciada. Essa ação faz com que os serviços da arquitetura carreguem a tarefa selecionada e a coloquem em execução, realizando as devidas chamadas entre o pEHS e o EXEHDA, além de manter o seu estado de execução.

Na parte direita superior está localizada a seção destinada à sugestão das tarefas inferidas de acordo com o estado atual do contexto. A atualização dessa seção é possível graças ao registro da IETC no Serviço de Predição de Tarefas, dessa forma, assim que uma tarefa é inferida, a interface é notificada para disponibilizá-la ao usuário.

Na Figura 10 é possível notar que o sistema inferiu a execução da tarefa "Receituário" sugerindo-a para a próxima execução a partir do histórico de execução do usuário em questão

[4] Trabalho em desenvolvimento por Alencar Machado. Integrante do GMob e mestrando do Programa de Pós-Graduação em Informática (PPGI) da Universidade Federal de Santa Maria.

e o estado do contexto capturado naquele momento. Caso o usuário opte por executar a tarefa inferida, ele deve clicar no botão "*Accept*" para que a sua execução se inicie.

O sistema tem a preocupação em não tomar decisões automáticas, mas sim, apresentar a tarefa inferida como auxílio na utilização do sistema. Dessa forma, o sistema não age se sobrepondo às decisões do usuário, interferindo negativamente no seu uso, mas apenas procura melhorar a usabilidade, identificando o próximo passo para a realização de uma atividade. Essa preocupação está de acordo com o resultado da pesquisa realizada no trabalho do Hospital do Futuro (BARDRAM, 2003).

4.3.4 Processo de Predição de Tarefas

A ideia inicial para a predição de tarefas foi construir grafos ligando os elementos do contexto e as tarefas do histórico de execução dos usuários, sem nenhuma forma de filtragem ou redução da quantidade de elementos. Essa abordagem se tornaria inviável uma vez que o histórico de utilização do sistema aumenta-se, levando a grafos com inúmeros nodos e arestas que, em muitos casos, não teriam real influência para a inferência em si. Diante disso, foram pesquisadas alternativas encontrando-se o mecanismo de árvore de decisão, constantemente empregado na mineração de dados.

A árvore de decisão é construída a partir dos estados dos elementos de contexto presentes no histórico de execução de um usuário. A Figura 11 exemplifica uma árvore de decisão construída para a realização da predição de tarefas a partir de dados de exemplo.

Na árvore, os nós principais são os elementos de contexto e as suas arestas são os possíveis valores dos seus estados, de acordo com as condições estabelecidas. As folhas da árvore representam a tarefa inferida após a sua navegação. Nesse caso, pode-se perceber que, caso o valor da propriedade "*Time Frame*" seja menor ou igual a "16,50", a única tarefa possível de ser inferida é "Receituário". Porém, se esse valor for maior que "16,50", deve-se levar em consideração o estado de outros elementos do contexto até que se chegue a uma tarefa, navegando-se em profundidade na árvore.

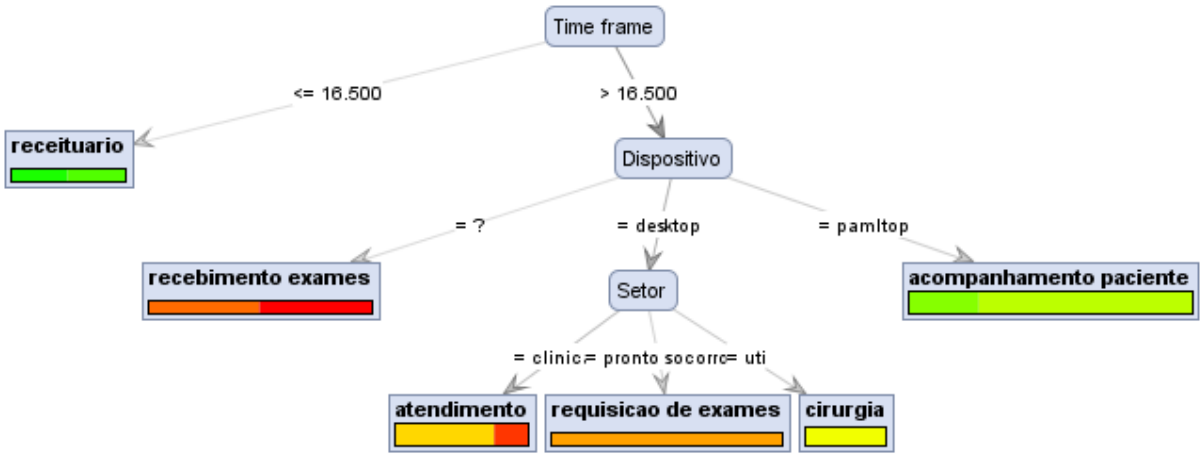

Figura 11 – Árvore de decisão

Para a construção da árvore, utilizou-se o *software* RapidMiner (MIERSWA, 2006), voltado à classificação de dados. O *software* permite a escolha de diversos algoritmos para a mineração de dados, podendo-se inclusive adicionar várias funcionalidades, como validação de dados, construção de gráficos, etc. Ainda, é possível que as informações sejam lidas e gravadas de diversas fontes, flexibilizando a sua utilização. Tais funcionalidades estão presentes em sua interface gráfica, porém sua API é disponibilizada para a inclusão em outros programas.

A construção e testes da árvore de decisão para a inferência de tarefas foram realizados na interface gráfica até que a validação estivesse completa. Para isso, foram geradas árvores de decisão com dados fictícios que, então, foram observadas e modificadas até que fossem condizentes com os resultados esperados.

Para a utilização das facilidades do RapidMiner, foi utilizada a sua API, na forma de objetos e arquivos de configuração no SPT. No momento em que um usuário executa uma nova tarefa a partir da interface, ele se registra no SPT para receber os eventos de predição de tarefas gerados. O SPT, por sua vez, constrói a árvore de decisão do usuário em questão, utilizando o algoritmo C4.5, de forma a ter condições de realizar a predição de tarefas baseada nos estados dos elementos de contexto.

Desenvolvido originalmente por Quinlan (QUINLAN, 1993), o algoritmo C4.5 realiza o processo chamado de inferência indutiva que é “o processo de partir de um modelo concreto de exemplos para modelos gerais” (QUINLAN, 1996), aplicável a outros conjuntos de dados para classificá-los. O C4.5 é uma melhoria do seu antecessor ID3 (*Interative Dichotomiser* 3 – “Dicotomizador” Interativo 3) (QUINLAN, 1986) visando uma melhor classificação dos dados, porém ambos utilizam árvores de decisão.

O algoritmo constrói um modelo padrão, a partir de dados de testes (ou dados de aprendizado), que é utilizado para a classificação de novos dados, sendo um dos mais aplicados nos dias de hoje para a mineração de dados (*data mining*) (WU, 2007).

A estrutura de dados utilizada é a árvore que é constituída de nós, contendo decisões e testes a serem executados, e folhas representado a classe a qual certo dado pertence. A partir do nó raiz da árvore, são feitos testes sobre os atributos dos dados de teste a fim de se navegar até uma folha com o mínimo de esforço necessário, porém utilizando os critérios mais corretos na classificação, de acordo com o princípio de Occam´s Razor[5].

No âmbito do projeto ClinicSpace, os atributos envolvidos são os dados do contexto captados e presentes no histórico de execução dos usuários. Dessa forma, os tipos de atributos podem ser: *time frame*, *sector, device*, *people around* e *devices around*. As classes dos dados, nesse caso, são as tarefas provindas do histórico do usuário. Assim, a navegação da árvore é feita realizando-se os devidos testes sobre os dados do contexto até que uma folha seja alcançada e, consequentemente, uma tarefa seja descoberta.

Caso todos os dados pertençam a uma única classe, a árvore é composta de apenas uma folha representando essa classe. Caso contrário, seleciona-se o atributo com o maior valor do cálculo de ganho de informação (*information gain*) que será usado como nó raiz e dividirá a árvore. Para cada possível valor resultante do teste, a árvore é construída recursivamente procurando-se o melhor teste a ser feito em cada nível até que as folhas sejam alcançadas. Ainda, os testes realizados podem ser feitos sobre dados nominais ou numéricos, quando deve ser aplicado um método para definir qual o melhor valor (*threshold*) a ser usado para dividir uma árvore.

Para a decisão sobre quais atributos melhor dividem os nós da árvore, primeiramente é calculada a entropia da informação do conjunto de dados como um todo, demonstrado na Equação 1.

$$E(S) = -\sum_{j=1}^{m} \frac{S_j}{S} log_2 \frac{S_j}{S}$$

Equação 1 – Cálculo da entropia da informação do conjunto de dados

O resultado dessa computação identifica a quantidade mínima de informação

[5] Princípio no qual “entidades não devem ser multiplicadas além do necessário”, ou seja, a menor explicação sobre uma determinada hipótese é a melhor, já que é melhor compreendida (KOHAVI, 1999).

necessária para a classificação de um dado baseado nos dados de exemplo (S). Para isso, é calculado o somatório da frequência de cada classe no conjunto como um todo ($\frac{S_j}{S}$) (KOHAVI, 1999) (HUANG, 2009).

Após o cálculo da entropia do conjunto de dados geral, para cada atributo é calculada a sua entropia, de forma a encontrar a frequência de cada atributo em cada classe presente no conjunto de dados, conforme demonstrado na Equação 2.

$$E(A) = -\sum_{j=1}^{m} \frac{S_1 + ... + S_j}{S} E(S)$$

Equação 2 – Cálculo da entropia da informação dos atributos (HUANG, 2009)

O ganho de informação da cada atributo é então calculado a fim de se achar qual dos atributos será efetivamente utilizado na divisão dos dados, demonstrada na Equação 3. O atributo com maior valor de ganho de informação constitui o teste da raiz da árvore formando as arestas para cada possível valor resultante desse teste. Para a formação do restante da árvore, o algoritmo é executado sucessivamente calculando-se as entropias de cada atributo sobre o conjunto de dados restante.

$$Ganho(A) = E(S) - E(A)$$

Equação 3 – Cálculo do ganho de informação dos atributos (HUANG, 2009)

Após a construção da árvore de decisão, o algoritmo realiza uma varredura para retirar nós que não contribuam efetivamente para a classificação dos dados. Esse processo, chamado *pruning* (poda), é realizado no sentido das folhas para a raiz em um único passo, analisando cada nó presente e levando em consideração uma taxa máxima de erro tolerada.

Caso a decisão de um determinado nó não ultrapasse essa taxa de erro tolerada, o algoritmo pode substituí-lo diretamente por uma folha com a classificação mais frequente no conjunto de dados de teste. Com isso, têm-se árvores de decisão com o menor tamanho possível, diminuindo o uso de memória e aumentando a eficiência durante a sua navegação.

Após a construção da árvore, utilizando o software RapidMiner, são construídos mapas[6] que possuem como chave o elemento do contexto a ser levado em consideração e

[6] HashMap: implementação de *hash tables* (tabelas hash) presente na API Java desde a versão 1.2. A última

como valor uma coleção de comparações sobre o elemento. A Figura 12 demonstra o mapa construído a partir da árvore de decisão gerada. Escolheu-se a utilização de um mapa devido ao seu desempenho nas operações de consulta que possui uma complexidade *O(1)* (LOUDON, 1999).

Esse desempenho é possível graças à utilização de uma função *hash* para a geração das chaves visando poucas colisões. Caso não existisse o mapa, toda a árvore deveria ser percorrida sempre que um elemento do contexto se modificasse até que uma condição fosse satisfeita, aumentando o tempo consulta e resposta ao usuário final.

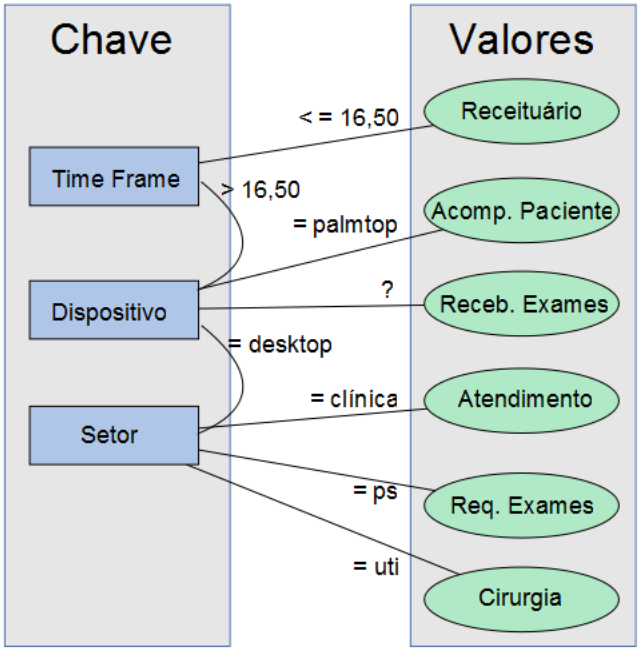

Figura 12 – Mapa dos elementos do contexto

Com base em um valor de um elemento de contexto, consulta-se o mapa para saber qual a atividade inferida. Caso o valor captado satisfaça uma condição que já especifique uma tarefa, essa é retornada para o serviço interessado na predição de tarefas. Caso a condição sobre o valor captado indique que a condição de outro elemento deve ser consultada, prossegue-se a iteração no mapa até que uma tarefa seja encontrada.

4.4 Serviço Spring

De modo a facilitar o desenvolvimento dos serviços necessários à inferência de tarefas, escolheu-se adotar a utilização do Spring *Framework* (WALLS; BREIDENBACH, 2008). Para isso, foi desenvolvido um serviço específico do qual os demais serviços são clientes: *ApplicationContextService*.

O Spring *Framework* provê funcionalidades necessárias ao desenvolvimento de aplicações corporativas, sem necessitar de um container JavaEE[7]. Entre tais funcionalidades, podem-se citar: injeção de dependência (*DI – Dependency Injection*), também conhecido como inversão de controle (*IoC – Inversion of Control*); gerenciamento de transações e uma extensa API para a programação de aplicações *web* (SPRING SOURCE, 2009).

Para a sua configuração são utilizados arquivos XML, nos quais são declaradas as classes necessárias ao sistema, além de configurar as dependências entre as próprias classes. O serviço *ApplicationContextService* desenvolvido para o EXEHDA, faz a leitura do arquivo de configuração do Spring durante a sua inicialização. Após essa leitura, os objetos criados ficam disponíveis para serem acessados pelos demais serviços presentes na arquitetura.

Os objetos utilizados para manipulação dos dados utilizam o padrão de projeto DAO (*Data Access Object* – Objeto de Acesso a Dados) (ALUR; CRUPI; MALKS, 2003) e possuem as operações básicas para a realização das operações CRUD (*Create, Retrieve, Update, Delete* – Criação, Busca, Atualização e Remoção) sobre as entidades utilizadas no sistema, são instanciados e configurados pelo *ApplicationContextService*. Dessa forma, a lógica envolvida na manipulação dos dados, realizada com apoio do framework Hibernate, permanece isolada do restante da arquitetura, facilitando a sua manutenção. O SHE é um dos clientes desse serviço, uma vez que na sua inicialização, os objetos necessários são disponibilizados através do Spring, tornando o desenvolvimento mais focado nas funcionalidades necessárias e abstraindo as configurações requeridas.

A configuração dos objetos do *framework* Hibernate, responsáveis pelo Mapeamento Objeto-Relacional das classes utilizadas no projeto para o Banco de Dados, é realizada pelo Spring, fazendo com que não seja necessário o desenvolvimento de serviços para tal finalidade. No arquivo XML são definidos os parâmetros de conexão ao Banco de Dados e demais configurações, como utilização de *caches* de objetos. A adoção dessa estratégia faz

[7] Java Enterprise Edition (SUN MICROSYSTEMS, 2008)

com que possíveis manutenções no Banco de Dados sejam facilitadas, pois o código-fonte das classes envolvidas permanece inalterado.

Para garantir a consistência dos dados persistidos pelo SHE, utiliza-se o controle de transações do Spring baseado em anotações. Nos métodos que fazem a persistência de dados, adicionou-se a anotação *@Transactional*, conforme a Figura 13, fazendo com que o Spring os detecte e introduza o devido nível de isolamento, garantindo as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), requeridas nesses casos.

```
@Transactional(readOnly=false, propagation=Propagation.NESTED)
public T save(final T entity) {
```

Figura 13 – Anotação de controle de transação

O desenvolvimento de novos serviços e classes, caso seja necessário à arquitetura ClinicSpace, poderá utilizar o *ApplicationContextService* para a construção e configuração de objetos. Isso faz com que a tarefa de implementação de novas funcionalidades torne-se mais prática e reduza o esforço necessário, uma vez que as configurações são feitas de maneira descritiva e, após a criação dos objetos, eles permanecem disponíveis ao restante do sistema.

5 RESULTADOS E DISCUSSÃO

Nesse capítulo são elencados os principais trabalhos existentes relacionados à área de inferência de tarefas, além de apresentar os resultados obtidos durante os testes simulados do sistema de inferência.

5.1 Testes e Avaliação dos Resultados

De forma a avaliar a solução desenvolvida, foram feitos testes a partir de dados sintéticos gerados pelo próprio sistema. A partir da análise dos dados foram identificados gargalos de desempenho e, com isso, foram realizadas as correções necessárias para que o processamento do sistema interfira o mínimo possível no desempenho das atividades pelos usuários.

5.1.1 Desempenho não-intrusivo

A análise de desempenho foi realizada em um computador com as seguintes características: Processador AMD Athlon^{tm} 64-bit Dual Core 4800+ e 4GB de memória RAM. Durante a captura dos dados, somente o sistema operacional, os *softwares* mínimos necessários e o Sistema Gerenciador de Banco de Dados permaneceram em execução, de forma a diminuir o impacto dos demais programas nos tempos de execução. A utilização do SGBD na mesma máquina da realização dos testes fez com que não houvesse interferência nos tempos necessários para a comunicação entre máquinas em rede, permitindo um melhor isolamento dos aspectos observados.

A Figura 14 exibe o gráfico de desempenho do sistema comparando a quantidade de itens presentes no histórico e o tempo necessário, em milissegundos, para a construção da árvore de decisão, principal estrutura de dados utilizada na inferência das tarefas.

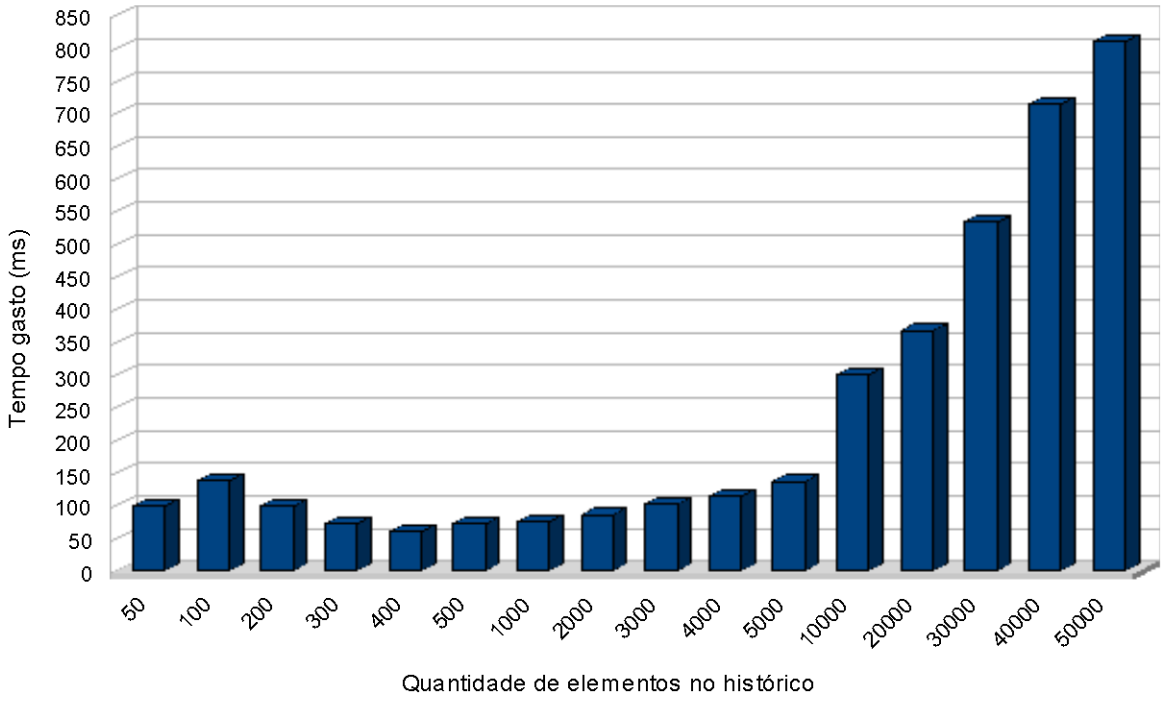

Figura 14 – Desempenho para a construção da árvore de decisão

A medição dos tempos foi realizada a partir do momento em que o usuário inicia a aplicação de disparo das tarefas até o momento em que a árvore de decisão está pronta para ser utilizada pelo sistema. Os dados de teste gerados foram inseridos no histórico de execução do usuário e foram incrementados de acordo com o progresso das medições. Para cada quantidade de elementos medida, foram realizadas dez execuções da interface gráfica de acionamento das tarefas, de modo a diminuir o impacto de eventuais picos de processamento do Sistema Operacional assim como para diminuir o impacto do processo de coleta de lixo[8] realizado pela Máquina Virtual Java (*Java Virtual Machine*)

Para a escolha dos limites de quantidades de elementos do histórico, adotou-se o tempo de médio de execução das tarefas de 10 min. e uma jornada de trabalho de 8 horas diárias ininterruptas. Portanto, a menor quantidade de elementos medida corresponderia a pouco mais de um dia trabalho e a maior quantidade medida corresponderia a mil dias de trabalho ou, seja, aproximadamente três anos de trabalho.

Na Figura 15, pode-se perceber que quando o usuário contém entre 50 e 5.000 elementos de histórico de execução, correspondendo a um e 100 dias de uso do sistema respectivamente, a construção da árvore de decisão leva em geral menos de 100

[8] *Garbage Collection* – processo automático no qual objetos que não estão mais em uso pelos programas são descartados e removidos da memória a fim de otimizar o seu uso (SUN MICROSYSTEMS, 2006).

milissegundos. Para o usuário final, esse tempo de resposta é imperceptível e não afeta a usabilidade do sistema. (NIELSEN, 1993).

Para a análise de desempenho, a partir de 10.000 elementos de histórico de execução, as medições foram feitas com incrementos de 10.000 elementos, até a quantidade máxima analisada. Nota-se que a partir desse momento, o sistema leva mais tempo para realizar a carga e análise dos dados para a construção da árvore de decisão. No entanto, mesmo o tempo mais alto identificado, quando o histórico de execução possui 50.000 elementos, o sistema levou menos de um segundo, o que ainda não prejudica a sua usabilidade, pois não interfere no fluxo de pensamento e concentração do usuário (NIELSEN, 1993).

5.1.2 Desempenho da Inferência de uma tarefa

Para se conhecer o desempenho do tempo de inferência de uma tarefa foram medidos os tempos necessários entre a observação da mudança de um elemento do contexto e a descoberta da tarefa inferida. Durante essa medição, assumiu-se que o usuário já estava utilizando a interface gráfica, portanto, descartou-se o tempo de inicialização do sistema, porque esta seria a sua rotina na utilização do ClinicSpace.

Para garantir um mínimo de interferência de processos do Sistema Operacional, bem como os processos internos da JVM, foram aguardadas 10 mudanças de contexto para a captura dos tempos. Com isso, visou-se a captura dos dados o mais real possível.

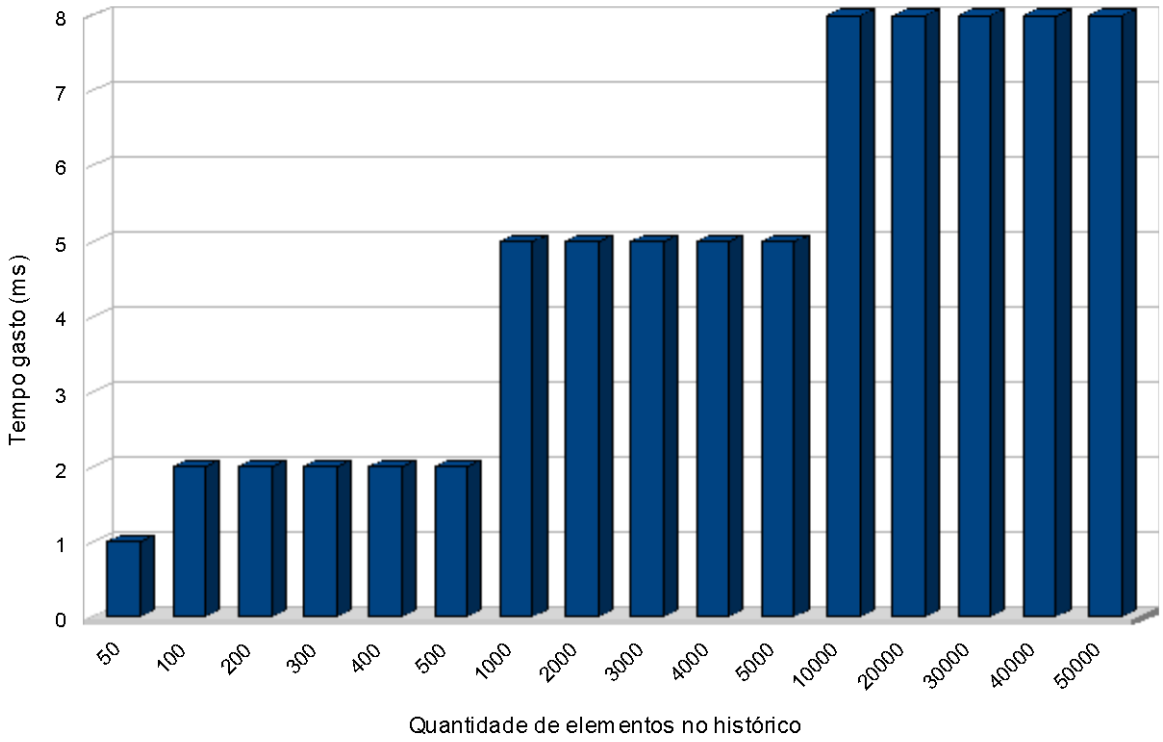

Figura 15 – Desempenho para a inferência de uma tarefa a partir do estado do contexto

A Figura 15 exibe o gráfico das capturas de tempos realizadas, comparando a quantidade de elementos nos histórico e o tempo gasto para percorrer o mapa de dados que representa a árvore de decisão encontrando uma tarefa. Pode-se notar que, para a menor quantidade de elementos, 50, o sistema necessita de apenas 1 milissegundo para encontrar a tarefa inferida. Para a quantidade de 50.000 elementos o sistema demora 8 milissegundos para realizar a inferência de uma tarefa com os dados capturados do ambiente.

Verifica-se, então, que o tempo necessário para identificar uma tarefa inferida não influencia de maneira significativa no desempenho do sistema como um todo. Mesmo o maior tempo identificado para a inferência de uma tarefa (8ms), é menor que 1% do tempo necessário para a construção e processamento da árvore de criação. Em termos de impacto no tempo de espera do usuário, esse processamento não é notado e não requer melhorias de desempenho.

A partir dos dados analisados, pode-se perceber que o usuário, durante a utilização do sistema dificilmente será afetado pelo processamento realizado para inferência. A construção da árvore de decisão e a descoberta de uma tarefa inferida necessitam de tempos que estão aquém do mínimo necessário para intervir na sua concentração e realização de tarefas (NIELSEN, 1993).

A verificação da eficiência do mecanismo de inferência, em si, não pôde ser comprovada, uma vez que a arquitetura ClinicSpace não está disponível ao usuário final. Dessa forma, a captura automática dos históricos de execução e, consequentemente, a sua utilização para a inferência de tarefas não foi possível.

Como esforço nesse sentido, foi iniciado um estudo objetivando construir formulários para a simulação de utilização do sistema que pudesse servir como base de dados do histórico de execução. No entanto, devido a dificuldades de interação com os usuários clínicos, os formulários não puderam ser plenamente refinados a ponto de servir como base para esse estudo, que então foi desconsiderado. Sugere-se, dessa forma, como futuro trabalho, a realização de medições das taxas de acerto do mecanismo de inferência a partir da disponibilização do ClinicSpace ao seu público-alvo.

5.2 Trabalhos Relacionados

O projeto Daidalos (KALATZIS, 2008) tem como foco fazer a inferência do contexto do usuário baseado no seu histórico de execução. Através do contínuo armazenamento do estado dos elementos de contexto presentes durante a execução de uma tarefa, o Daidalos pretende inferir o estado de elementos de contexto que, em um determinado momento, estejam inativos ou inoperantes.

Outra característica do Daidalos é a preocupação de distribuição dos componentes da arquitetura. Os nodos que executam o sistema enviam para o servidor o estado dos seus sensores de tempos em tempos e esse, por sua vez, os armazena para construir o histórico de execução do usuário. Em um processamento assíncrono, o servidor constrói tabelas de predições com *scores* das probabilidades de cada cenário se realizar novamente.

O nodo, assim que acessa pela primeira vez o servidor, recebe a tabela de probabilidades do contexto. Dessa forma, cada nodo possui uma tabela de predição do contexto e é capaz de inferir o estado de algum elemento de contexto. Há que se ressaltar que o objetivo do Daidalos é fazer a inferência sobre os elementos do contexto, tendo como principal métrica a inferência da localização geográfica do usuário e não as atividades que o usuário pode executar.

O sistema desenvolvido por Lester (LESTER, 2005) tem como objetivo reconhecer as atividades humanas diárias através da utilização de sensores introduzidos na roupa (*wearable*

sensors). Entre os sensores utilizados estão: acelerômetro, microfone, sensores de luz, pressão atmosférica, umidade do ar, temperatura ambiente e bússola.

Primeiramente, os hábitos das pessoas são observados durante certo período de tempo e, após, são usados modelos matemáticos para identificar quais sensores têm mais influência no reconhecimento das atividades. Após a seleção dos sensores, os dados capturados pelos sensores são utilizados no treinamento dos algoritmos selecionados, de maneira a aumentar as taxas de acertos. Evidencia-se que o objetivo aqui é identificar as tarefas humanas, necessitando que o sistema seja treinado sob um conjunto de dados, o que é desnecessário para o tipo de inferência desenvolvido no presente trabalho.

O projeto PROACT (*Probabilistic Activity Toolkit*) é um projeto para identificar e inferir atividades do dia-a-dia (ADL – *Activities of Daily Living*) (PHILIPOSE, 2003). Para realizar o seu propósito, a casa do usuário observado e os seus utensílios devem possuir etiquetas e leitores para a tecnologia RFID[9], proporcionando um ambiente menos intrusivo.

Os usuários monitorados devem preencher um formulário, especificando qual atividade está executando para que o sistema possa fazer a conexão com os dados coletados dos sensores. As atividades são decompostas em passos para a sua realização e, para cada passo, um determinado número de utensílios é usado. A partir dos passos das atividades e as informações dos sensores, modelos matemáticos são utilizados para determinar qual a próxima atividade a ser executada pelo usuário.

Nota-se que o objetivo geral do projeto PROACT é prover meios para que as pessoas que cuidam de pacientes ou doentes possam antecipar as tarefas e ajuda-las, facilitando as suas vidas. Ainda, caso o usuário tenha começado uma determinada atividade e tenha despendido mais tempo que o de costume, pode-se acionar algum mecanismo de emergência aumentando as chances de ajuda. O foco de aplicações é em cuidados de saúde em casa (*HomeCare*).

O trabalho desenvolvido por Gassen (GASSEN, 2010) tem por objetivo realizar a inferência de tarefas e aplicações do usuário em um ambiente hospitalar baseado em ontologias. Para isso, devem ser construídas as ontologias para cada tipo de ambiente, especificando-se quais situações presentes nos ambiente devem ser levadas em consideração para a pró-atividade do sistema.

Na ocorrência de mudanças no ambiente de execução da aplicação (contexto), são

[9] *Radio Frequency Identification* – Identificação por rádio freqüência. Tecnologia que insere, de maneira miniaturizada, um transponder de rádio em pequenas etiquetas, facilitando o seu uso (WIKIPEDIA, 2009).

executadas consultas no formato SQWRL[10] sobre as ontologias previamente construídas. O resultado dessas consultas pode ser apresentado como sugestão de aplicação a ser executada pelo usuário ou alguma outra ação previamente programada. Dessa forma, a inferência de tarefas, nesse caso, só pode ocorrer após uma prévia construção das ontologias dos possíveis ambientes de execução do sistema. Em caso de uma eventual remodelagem física, do ambiente, por exemplo, as ontologias afetadas devem ser revistas e adaptadas para refletir a nova realidade.

A Tabela 1 exibe uma comparação entre os projetos relacionados e o projeto desenvolvido neste trabalho.

Projeto	**Orientação a tarefas humanas**	**Configuração automática do sistema**	**Auto-adaptação do sistema (dinamicidade)**	**Inferência centralizada**
Daidalos	X	X	X	
PROACT				X
Lester	X			X
Gassen	X	X		X
ClinicSpace	X	X	X	X

Tabela 1 – Tabela comparativa entre projetos com foco em inferência

Para a realização da comparação entre os projetos foram usados os seguintes critérios:

a. Orientação a tarefas humanas: define se o sistema busca mapear os seus processos com base na execução humana de atividades/tarefas;
b. Configuração automática do sistema: especifica se o sistema deve ser pré-configurado pelo próprio usuário ou desenvolvedor para que o mecanismo de inferência funcione ou é prontamente disponível para uso. Entende-se que a presença desse fator é positiva ao sistema;
c. Auto-adaptação do sistema (dinamicidade): trata da questão na qual o sistema detecta a mudança dos hábitos de execução do sistema/ambiente ou necessita uma re-configuração;

[10] Linguagem de pesquisa sobre OWL (*Web Ontology Language* – Linguagem de Ontologias Web) (O´CONNOR, 2009).

d. Inferência centralizada: define se o sistema realiza a inferência em um ponto central do sistema ou descentralizada. Devido à natureza limitada da maioria dos dispositivos móveis atuais, deve-se evitar a sua sobrecarga com cálculos, centralizando-os em servidores com configurações específicas para esse fim. Dessa forma, entende-se que a inferência realizada de maneira centralizada é uma característica favorável aos sistemas.

6 CONCLUSÕES

A Computação Pervasiva, quando aplicada em um ambiente hospitalar tem a possibilidade de melhorar a disponibilização das informações e, com isso, tende a melhorar a forma de trabalho de seus usuários. O acesso ao sistema em diversas localizações permite uma maneira prática e rápida de realizar as tarefas cotidianas interferindo o mínimo possível na forma de trabalho de cada usuário.

De forma a agregar inferência de tarefas na arquitetura do projeto ClinicSpace, foram desenvolvidos componentes que armazenam as tarefas executadas pelo usuário juntamente com o estado do contexto presente nesse momento. O mecanismo de inferência utiliza esse conjunto de dados (histórico de execução de tarefas e contexto) para disponibilizar ao usuário uma sugestão sobre a próxima tarefa a ser executada. Fica evidente que essa abordagem possui um melhoramento contínuo, pois os hábitos do usuário fornecem os dados para a construção das árvores de decisão, estruturas de dados utilizadas para inferir as tarefas.

A detecção da mudança do estado dos elementos do contexto é utilizada para detectar a futura tarefa a ser executada pelo usuário. Dessa forma, o ambiente afeta o comportamento do sistema, e esse utiliza o ambiente para prover uma nova funcionalidade ao usuário.

As tarefas inferidas pelo sistema sempre são exibidas como sugestões na interface do usuário, sem a tomada de decisões automáticas. Com isso, espera-se melhorar a usabilidade do sistema, pois o fluxo de trabalho do usuário é sugerido e ele permanece com o controle total do sistema, em tempo integral.

6.1 Contribuições

Atualmente, a maioria dos trabalhos com foco em predição de tarefas humanas relaciona-se a casas inteligentes (*smart houses/homes*) ou à predição do comportamento de redes de computadores. Diferentemente, este trabalho foca na predição de tarefas que

compõem as atividades humanas voltado para um ambiente clínico, tendo como objetivo o auxílio ao usuário final.

O mecanismo de inferência das tarefas desenvolvido utiliza o histórico de execução do próprio usuário, fazendo com que os seus próprios hábitos modelem o sistema. Assim, o usuário não precisa ajustar ou configurar o sistema para que o mesmo realize a inferência. Além disso, o sistema não exige que o usuário realize uma seqüência de tarefas para treinar o algoritmo de inferência.

Dentro os projetos focados em inferência de atividade pesquisados, a maioria (i) coleta dados estaticamente, (ii) processa-os com algum modelo para depois ter condições de (iii) inferir as atividades dos usuários. Uma falha pode acontecer em tais modelos nas situações em que o usuário muda os seus hábitos, pois a modelagem da inferência foi realizada sobre dados de modelo que não refletem mais a realidade de uso do sistema, gerando sugestões de tarefas possivelmente inúteis. Construir as árvores de decisão baseadas no histórico de execução do usuário faz com que as atividades que não são mais executadas sejam descartadas pelos novos hábitos do usuário, adaptando-se ao seu perfil de utilização.

6.3 Publicações Realizadas

WIM 2010 – "Inferência de Atividades Clínicas a partir de Propriedades do Contexto". In: X Workshop de Informática Médica (WIM), XXX Congresso da Sociedade Brasileira de Computação.

CBIS 2010 – "Detecção de Atividades Clínicas a partir do Histórico de Execução do Usuário". In: XII Congresso Brasileiro de Informática na Saúde (CBIS) (aceito).

6.4 Trabalhos Futuros Sugeridos

Como trabalhos futuros sugerem-se testes e análises do mecanismo de inferência em um ambiente mais próximo da realidade dos usuários clínicos, após a disponibilização da Interface de Edicão de Tarefas e Contexto e a possibilidade de coleta de dados reais pela

arquitetura ClinicSpace. Dessa forma, diante dos hábitos cotidianos dos usuários, será possível ter uma real avaliação das inferências realizadas, possibilitando ajustes que possam melhorar a usabilidade do sistema. Ainda, a utilização do sistema em um ambiente real poderá indicar, por exemplo, qual a quantidade de dias do histórico de execução é realmente influente para a inferência, melhorando a sua eficiência e o desempenho geral do sistema.

Cabe ressaltar aqui que, conforme a utilização do sistema aumenta, a quantidade de itens do histórico de execução começa a se expandir, consequentemente, a construção da árvore de decisão para o usuário necessita de mais tempo, tornando o tempo de resposta ao usuário inaceitável. Fica evidente, contudo, que um estudo mais detalhado deve ser realizado a fim de estipular a quantidade de itens a ser levada em consideração para a inferência de tarefas, de acordo com o tempo de processamento necessário e a taxa de acertos das tarefas corretamente inferidas.

REFERÊNCIAS

ABOWD, G. D.; MYNATT, E. D. ***Charting Past, Present and Future Research in Ubiquitous Computing.*** In: ACM *Transactions on Computer-Human Interaction* (TOCHI), n. 1, vol. 7, 2000, USA. p. 29–58.

ALUR, D.; CRUPI, J.; MALKS, D. ***Core J2EE Patterns****: Best Practices and Design Strategies*. 2nd ed. California, USA: 2003. Sun Microsystems Press.

AUGUSTIN, I. **ClinicSpace**: auxílio às tarefas clínicas em um ambiente hospitalar do futuro baseado em tecnologias da Computação Ubíqua/Pervasiva. Projeto CNPQ, 2008-11.

AUGUSTIN, I.; SILVA, F.; RIZZETTI, T.; FERREIRA, G. G. L. **ClinicSpace:** ferramenta-piloto que permite aos clínicos a personalização de suas tarefas diárias (submetido a publicação).

AUGUSTIN, I,; YAMIN, A.; GEYER, C. F. R. ***Managing the follow-me semantics to build large-scale pervasive applications***. In: *Proceedings of the 3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing*, 2005, France, p. 1-8.

BARDRAM, J. E. ***Hospitals of the Future***: *Ubiquitous Computing Support for Medical Work in Hospitals*. In: *Proceedings of the 5th International Conference in Ubiquitous Computing*, 2003, USA.

BARDRAM, J. E.; CHRISTENSEN, H. B. ***Pervasive Computing Support for Hospitals***: *An overview of the Activity-Based Computing Project*. IEEE Pervasive Computing, vol. 6, issue 1, 2007, p. 44-51.

BAUER, C.; KING, G. ***Java Persistence with Hibernate***. 2nd ed. New York, USA: 2007. Manning Publications Co.

CHANDLER, P.; SWELLER, J. ***Cognitive Load Theory and the Format of Instruction***. In: *Cognition and Instruction*, n. 4, vol. 8, 1991. p. 293 – 332.

COUTAZ, J.; CROWLEY, J. L.; DOBSON, S.; GARLAN, D. ***Context is Key***. In: Communications of ACM, n. 3, vol. 48, 2008, USA. p. 49-53.

DADAM, P.; REICHERT, M.; KUHN, K. ***Clinical Workflows:*** *The Killer Application for Process-Oriented Information Systems ?* In: Proceedings of the 4th International Conference on Business Information Systems, 2000, Poland. p. 36-59.

DEY, A. K. ***Understanding and Using Context***. In: Personal and Ubiquitous Computing, vol. 5, n. 1. London, UK, 2001. p. 4 – 7.

FERREIRA, G. G. L. **Adicionando ao *Middleware* EXEHDA o Suporte a Aplicações Orientadas a Atividades Humanas Cotidianas**. Dissertação de Mestrado. Universidade Federal de Santa Maria, Santa Maria, RS, Brasil, 2009.

FERREIRA, G. L. et al.. ***Middleware for Management of End-user Programming of Clinical Activities in a Pervasive Environment***. In: 2009 *Workshop on Middleware for Ubiquitous and Pervasive Systems (WMUPS), Fourth International Conference on Communication System Software and Middleware*, 2009, Ireland. p.7-12.

FERREIRA, G. L.; SILVA, F. L.; LIBRELOTTO, G. R.; AUGUSTIN, I. **Adaptando o *Middleware* EXEHDA para o Tratamento de Atividades Clínicas**. In: XXXV Conferencia Latinoamericana de Informática (CLEI), Pelotas, RS, Brasil,.2009. p. 37-37.

GAMMA, E.; et al. ***Design Patterns****: Elements of Reusable Object-Oriented Software*. 1st ed. Massachussets, USA: 1995. Addison Wesley (*Professional Computing Series*)

GASSEN, J. B. **Uma Metodologia para o Uso de Ontologias Aplicadas à Descrição de Contexto em Ambientes Hospitalares Pervasivos**. Dissertação de Mestrado. Centro Universitário Franciscano, Santa Maria, RS, Brasil, 2010.

HUANG, Y.; HOA, V. T. T. ***General Criteria on Building Decision Trees for Data Classification***. In: *Proceedings of the 2nd International Conference on Interation Sciences* (ICIS), 2009, USA. p 649-654.

KALAPRIVA, K., et al. ***A Framework For Resource Discovery In Pervasive Computing For Mobile Aware Task Execution***. In: *Proceedings of the 1st conference on Computing Frontiers*, 2004, USA. p. 70-77.

KALATZIS, N., et al. ***User-centric Inference Based on History of Context Data in Pervasive Environments.*** In: *Proceedings of the 3rd International Workshops on Services Integration in Pervasive Environments*. p. 25-30. Italy: 2008.

LAERUM, H.; FAXVAAG, A. ***Task-Oriented Evaluation of Electronic Medical Record Systems:*** *Development and Validation of a Questionnaire for Physicians*. BMC Medical Informatics and Decision Making. 2004.

LESTER. J. et al. ***A Hybrid Discriminative/Generative Approach for Modeling Human Activities.*** In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence.* Edinburgh, Scotland: 2005. p. 766-772.

LOUDON, K. ***Mastering Algorithms with C***. 1st ed.California, USA: 1999. O´Reilly Media, Inc.

MACHADO, A.; LIBRELOTTO, G. R.; AUGUSTIN, I. **Ferramenta para Definição de Contexto pelo Usuário Final na Programação de Tarefas Clínicas em Sistema de Saúde Pervasivo**. In: II Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP), Belo Horizonte, MG, Brasil, 2010. p.316-325.

MAYRHOFER, R. M. ***An Architecture for Context Prediction***. Tese de Doutorado. Áustria, 2004.

MIERSWA, I, et al. ***Yale (now RapidMiner)****: Rapid Prototyping for Complex Data Mining Tasks*. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, Philadelphia, PA, USA. p. 935-940.

NIELSEN, J. ***Usability Engineering.*** 1st ed. California, USA: 1993. Morgan Kaufmann.

NURMI, P.; MARTIN, M.; FLANAGAN, J. A. ***Enabling Proactiveness through Context Prediction***. In: *Proceedings of the Workshop on Context Awareness for Proactive Systems.* Helsinki, Finland: 2005. p. 159-168.

O´CONNOR, M.; DAS, A. **SQWRL**: *a Query Language for OWL*. In: *Proceedings of OWL: Experiences and Directions Workshop* (OWLED), 2009, USA.

QUINLAN, J. R. ***Induction of Decision Trees***. In: *Machine Learning*, n. 1, vol. 1, 1986, USA. p 81-106.

QUINLAN, J. R. **C4.5**: *Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993, San Francisco, CA, USA.

QUINLAN, J. R. ***Learning Decision Tree Classifiers***. In: ACM Computing Surveys (CSUR), n. 1, vol. 28, 1996, USA. p 71-72.

PHILIPOSE, M.,et al. ***The Probabilistic Activity Toolkit:*** *Towards Enabling Activity-Aware Computer Interfaces*. Intel Research Seattle Tech Memo. Seattle, USA: 2003.

RAGGETT, D. ***The Ubiquitous Web***, 2005. Disponível em: <http://www.w3.org/2005/Talks/0621-dsr-ubiweb/#(1)>. Acesso em: 4 abr. 2008.

RIZETTI, T. ***Framework*** **para gerenciamento e personalização de contexto orientado a tarefas.** Dissertação de mestrado. Universidade Federal da Santa Maria, Santa Maria, RS, Brasil, 2009.

SCHILIT, B. N., ADAMS, N., WANT, R. ***Context-Aware Computing Applications***. In: IEEE Workshop on Mobile Computing Systems and Applications, 1994, USA.

SILVA, F. S. **ClinicSpace**: Modelagem de uma Ferramenta-Piloto para Definição de Tarefas Clínicas em um Ambiente de Computação Pervasiva Baseado em Tarefas e Direcionado ao Usuário-Final. Dissertação de Mestrado. Santa Maria, RS, Brasil, 2009.

SILVA, F. L. et al. **Ferramenta para a Programação pelo Usuário-Final de Tarefas Clínicas em um Ambiente de Saúde Ubíquo**. In: XXXV Conferência Latinoamericana de Informática (CLEI), Pelotas, RS, Brasil, 2009. p. 37-37.

SPRING SOURCE. ***Spring Framework:*** Reference Documentation, 3.0. [S.l.]: 2009. Disponível em: <http://static.springsource.org/spring/docs/3.0.x/spring-*framework*-reference/html/>. Acesso em: 20 dez. 2009.

SUN MICROSYSTEMS. ***The Java ME Platform:*** *The Most Ubiquitous Application Platform for Mobile Devices.* Disponível em: <http://java.sun.com/javame/>. Acesso em: 4 abr. 2008.

SUN MICROSYSTEMS. ***Memory Management in the Java HotSpot^{tm} Virtual Machine***. [S.l.], 2006 Disponível em: <http://java.sun.com/javase/technologies/hotspot/gc/memorymanagement_whitepaper.pdf>. Acesso em 22 jul. 2010.

SUN MICROSYSTEMS. ***The JavaEE Tutorial For Sun Java System Application Server 9.1.*** Santa Clara, CA, USA, 2008. Disponível em: <http://download.oracle.com/docs/cd/E17477_01/javaee/5/tutorial/doc/javaeetutorial5.pdf>. Acesso em: 22 jul. 2010.

VICENTINI, C. **PEHS**: Arquitetura de um Sistema Pervasivo de Informação em Saúde Orientado às Atividades Personalizadas pelo Usuário Clínico. Dissertação de mestrado. Santa Maria, RS, Brasil, 2010.

VICENTINI, C., MACHADO, A., AUGUSTIN, I. **Requisitos de um Registro Eletrônico de Saúde Ubíquo.** In: VIII Simpósio de Informática da Região Centro/RS (SIRC/RS). Santa Maria, RS, Brasil, 2009.

WALLS, C.; BREIDENBACH, R. ***Spring in Action***. 2nd ed. New York, USA: 2008. Manning Publications Co.

WEISER, M. ***Some Science Computer Issues in Ubiquitous Computing***. In: Communications of the ACM, n. 7, vol. 36, 1993, USA. P. 75-84.

WEISER, M. ***The Computer for the 21st Century.*** In: Scientific American, n. 3, vol. 265, 1991, USA. p. 94-104.

WIKIPEDIA. **RFID**: *Radio Frequency Identification*. [S.l.]: [200-]. Disponível em: <http://pt.wikipedia.org/wiki/RFID>. Acesso em 20 dez. 2009.

WORKFLOW MANAGEMENT COALITION. ***Terminology & Glossary*****. Hampshire, UK, 1999. V. 3.0**

WU, X, et al. ***Top 10 Algorithms in Data Mining.*** In: Knowledge and Information Systems, n. 1, vol. 14, 2007, New York, NY, USA. p.1-37.

YAMIM, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva.** Tese de Doutorado. Porto Alegre, RS, Brasil, 2004.

APÊNDICE A – Diagrama de Classes de Serviços

No presente Apêndice é apresentado o Diagrama de Classes dos serviços desenvolvidos para a arquitetura de inferência de tarefas.

Na Figura 16 são apresentadas as classes dos serviços envolvidos. A principal interface do sistema é *PredictionService*, que representa o serviço que efetivamente realiza a predição, possuindo os métodos "*register*" que permitem o registro de componentes para o recebimento de eventos de inferência. De forma inversa, os métodos "*unregister*" cancelam os eventos de inferência para um determinado componente.

A classe *DecisionTree*, utilizada pela interface *PredictionService*, representa a árvore de decisão que é utilizada para a inferências das tarefas. Essa classe possui métodos para a construção da árvore em si a partir dos objetos da API do *RapidMiner*. Além disso, a árvore possui um método que retorna a provável tarefa a partir do estado atual do contexto.

A interface *ExecutionHistoryService* tem a responsabilidade de captar o contexto no início e no término da execução de uma tarefa e armazená-lo no Banco de Dados. As operações envolvendo o acesso do histórico são realizadas com a utilização da interface *HistoryService*, que por sua vez utiliza os serviços *UserService* e *TaskService* para acesso aos usuários e tarefas armazenados, respectivamente. A interface *Service* possui métodos utilitários para a manipulação dos dados, como, por exemplo, "*save*" e "*update*", e é estendida pelas interfaces *UserService* e *TaskService*.

A interface *ServicoGerenciamentoTarefas* é responsável por instanciar as tarefas solicitadas pelo usuário e utiliza a interface *ExecutionHistoryService*. Dessa maneira, no momento que uma tarefa inicia, os dados do contexto são captados e armazenados para as futuras inferências. O momento de término da tarefa também é armazenado, para fins de cálculo de duração das tarefas.

A interface *ApplicationContextService* realiza a leitura dos arquivos de configuração do *framework* Spring e disponibiliza os objetos (*beans*) para os demais serviços. Como exemplo, o serviço *HistoryService* é acessado pelo *ExecutionHistoryService* através do service *ApplicationContextService*.

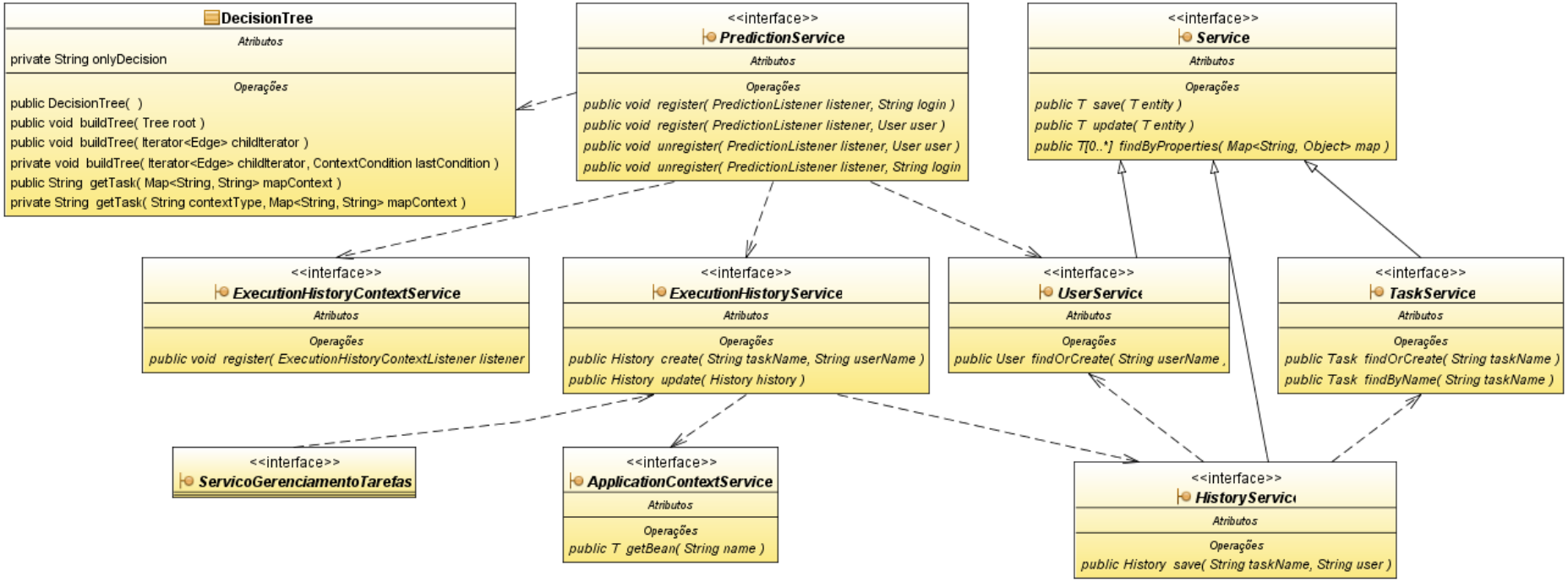

Figura 16 - Diagrama de classes dos serviços

APÊNDICE B – Diagrama de Classes de Entidades

Nessa seção é apresentado o Diagrama de Classes das entidades utilizadas para representar as informações necessárias à arquitetura de inferência de tarefas desenvolvida.

A Figura 17exibe as entidades utilizadas na arquitetura como um todo e as demais entidades envolvidas para a realização da inferência de tarefas.

A classe *User* representa os usuários do sistema e possui relacionamento com a classe *Role* que especifica os papéis do usuário no sistema, podendo ser Médico, Enfermeiro, Técnico de Enfermagem, etc. Dessa forma, pode-se conhecer o usuário que está executando o sistema, juntamente com a sua função no seu ambiente.

A classe *Task* representa uma tarefa presente no sistema e que pode ser disponibilizada ao usuário para execução. A classe *History* representa o histórico de execução das tarefas de um determinado usuário. A classe possui propriedades que identificam o momento de início e fim da realização de uma atividade, além de possuir relacionamentos com as classes *HistoryContext* e *SecundaryHistoryContext*.

A classe *HistoryContext* possui as informações necessárias para armazenar o contexto de execução de uma determinada tarefa, sendo ligada ao histórico dessa mesma tarefa. Nessa classe, pode-se perceber a presença de propriedades que armazenam o dispositivo, a localização (propriedade *sector*) e o *time frame*. Propriedades essas captadas no momento da realização de uma tarefa e armazenadas para a realização da inferência de tarefas.

A classe *SecundaryHistoryContext* armazena as demais informações captadas do contexto durante a realização de uma tarefa, possuindo propriedades que identificam o tipo de informação e o valor capturado pelos sensores. Como exemplo, se no momento de realização da tarefa for identificado um aparelho RX, a propriedade *type* possuirá o valor ‘devices’ e a propriedade *contextInfo* possuirá o valor ‘RX’. Dessa forma, tais informações podem ser utilizadas na inferência das tarefas.

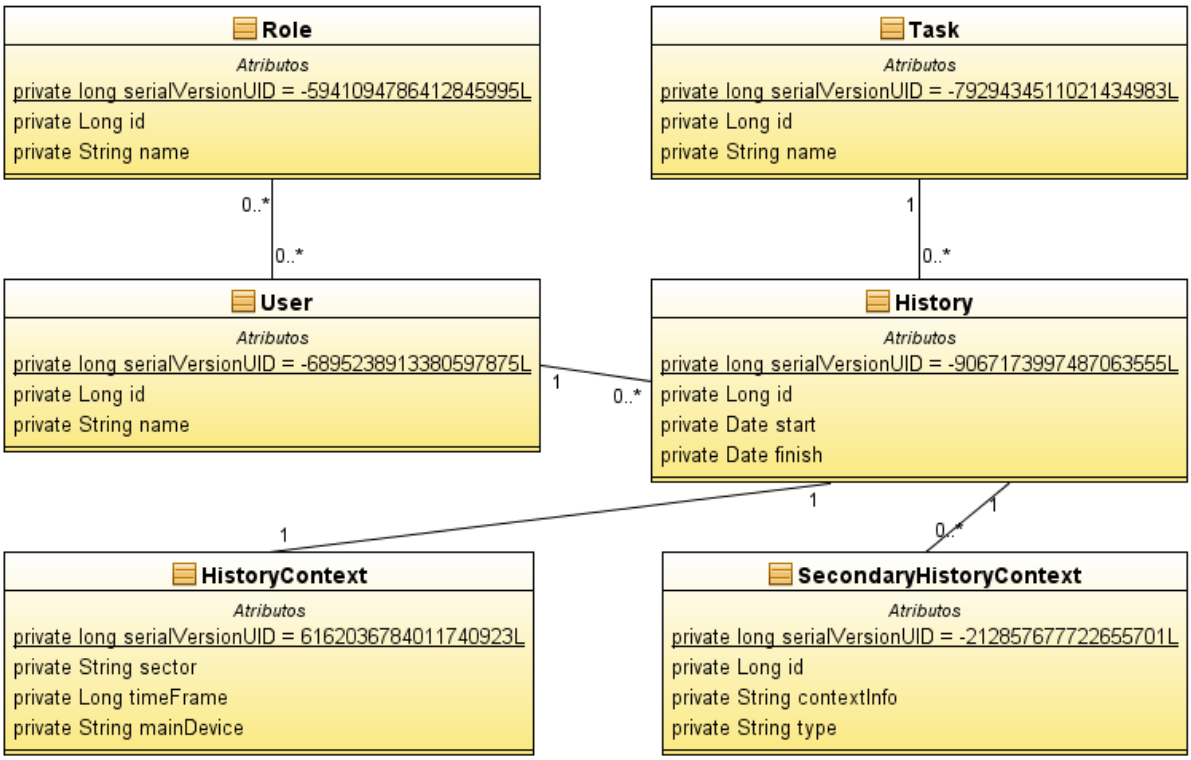

Figura 17 - Diagrama de classes das entidade